A CONNECTIONIST/CONTROL ARCHITECTURE FOR WORKING MEMORY

Technical Report AIP - 23

Walter Schneider & Mark Detweiler

Learning Research and Development Center
and Psychology Department
University of Pittsburgh
Pittsburgh, PA. 15260

The Artificial Intelligence and Psychology Project

Departments of
Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh

A CONNECTIONIST/CONTROL ARCHITECTURE FOR WORKING MEMORY

Technical Report AIP - 23

Walter Schneider & Mark Detweiler

Learning Research and Development Center
and Psychology Department
University of Pittsburgh
Pittsburgh, PA. 15260

29 September 1987

This research was supported by the Computer Sciences Division, Office of Naval Research and DARPA under Contract Number N00014-86-K-0678; the Army Research Institute under Contract Number MDA903-86-C-0149 and Personnel and Training Research Programs, Psychological Sciences Division, ONR Contract Number N-0014-86-K-0107. Reproduction in whole or in part is permitted for purposes of the United States Government. Approved for public release; distribution unlimited.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AIP - 23	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Computer Sciences Division Office of Naval Research (Code 1133)
6c. ADDRESS (City, State, and ZIP Code) Department of Psychology Pittsburgh, Pennsylvania 15213	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, Virginia 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Same as Monitoring Organization 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0678
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS p400005ub201/7-4-86 PROGRAM ELEMENT NO: N/A; PROJECT NO: N/A; TASK NO: N/A; WORK UNIT ACCESSION NO: N/A
11. TITLE (Include Security Classification) A CONNECTIONIST/CONTROL ARCHITECTURE FOR WORKING MEMORY	
12. PERSONAL AUTHOR(S) Walter Schneider & Mark Detweiler	
13a. TYPE OF REPORT Technical 13b. TIME COVERED FROM 86Sept15 TO 91Sept14	14. DATE OF REPORT (Year, Month, Day) 87 September 29 15. PAGE COUNT 34
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB-GROUP	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Artificial intelligence, connectionism, cognitive psychology, working memory, long-term memory, short-term memory

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

SEE REVERSE SIDE

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan L. Meyrowitz	22b. TELEPHONE (Include Area Code) (202) 696-4302 22c. OFFICE SYMBOL N00014

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

ABSTRACT

A connectionist/control architecture and simulation are described. The model is detailed at three levels of scale. The system-scale includes regions that specialize in different classes of processing. The activity of the regions is coordinated by a central control structure that routes control signals among regions and sequences transmissions among regions to limit message interference. One region serves as a context storage mechanism that can reactivate messages contained on a innerloop of processing. At the macro scale, each region is divided into a number of levels that sequentially or spatially input or output the patterns to other levels. Each level has a control structure that monitors the activity of all the modules in its level and controls the signals to coordinate the sequential storage and processing of information. At the micro scale, each level includes multiple modules. Each of these modules involves a connectionist network that processes vectors of information. A module can store, categorize, maintain, and prioritize a received vector. This architecture provides an interpretation of working memory phenomena including the magic number 3 or 4, acoustic confusions, sequential processing, problems with digit cancelling, difficulty maintaining order information, elaborative versus maintenance rehearsal, episodic versus semantic memory, release from proactive interference, long-term memory recency effects, robut processing during short-term memory overload, and proactive and retroactive effects.

A CONNECTIONIST/CONTROL ARCHITECTURE FOR WORKING MEMORY

Walter Schneider
Mark Detweiler

LEARNING RESEARCH AND DEVELOPMENT CENTER
AND PSYCHOLOGY DEPARTMENT
UNIVERSITY OF PITTSBURGH
PITTSBURGH, PENNSYLVANIA 15260

THE PSYCHOLOGY OF LEARNING AND MOTIVATION, VOL. 21

I. Introduction

The years since the mid 1950s have witnessed a number of important movements in the study of short-term memory (STM). Miller (1956) introduced the concept of capacity limits of STM, citing the magic number 7 ± 2; Broadbent (1958) drafted the first serious information-processing model of STM; and Brown (1958) and Peterson and Peterson (1959) rediscovered the technique of using an interpolated task to prevent rehearsal over a brief retention interval. In the 1960s, Melton (1963) advanced the view of interference as the source of all forgetting, Keppel and Underwood (1962) demonstrated the reality of proactive inhibition, and Waugh and Norman (1965) and Atkinson and Shiffrin (1968) proposed what must now be regarded as the modal models of STM. In the 1970s, Baddeley and Hitch (1974) developed and elaborated the idea of a working memory system. And in the 1980s research and theory building are continuing to further differentiate the phenomena and mechanisms behind working-memory systems, e.g., Baddeley (1986) and Chase and Ericsson (1981, 1982).

Since the mid 1970s, the modal model of STM has come under increasing criticism (Baddeley, 1976, 1986; Crowder, 1982; Klapp, Marshburn & Lester, 1983; Klapp, 1987). STM capacity appears more variable than Miller suggested, varying from size 2 in digit canceling (Kirchner, 1958) to size 80 in a skilled memory task (Chase & Ericsson, 1981). More importantly, most real-world tasks could not be completed if working memory had only five to nine items. For example, production system models such as ACT* (J. R. Anderson, 1983) used to simulate real-world tasks typically require a working memory of 20 items to maintain variable information and the goal stack. Further, consider a task such as electronic troubleshooting. To troubleshoot effectively one must at any point have in working memory the fault, the good state, the position in the circuit, the critical input and output signals, the expected signal, and the current hypothesis. If technicians are temporarily interrupted while tracing a fault, they do not not have to start all over. After a few seconds, they continue as if the interruption had never occurred.

As a final difficulty of capacity-limited theories of STM, consider that practitioners interested in human workload have long sought to identify the "red line" at which performance undergoes catastrophic failure, e.g., air-traffic controllers being interrupted and completely losing their ability to direct air traffic. Such failures are very rare. Humans tend to become slower and somewhat more error prone with increases in task loading, but there is no "red line," and catastrophic failures do not appear suddenly when the task requires remembering more than seven chunks of information. In other words, human performance shows "graceful degradation" in situations of memory overload.

In this article we trace some of these developments and offer a view of working memory situated within a 1980s connectionist framework. We also discuss a number of phenomena which do not fit neatly into the textbook treatments of the modal model. And while we nevertheless endorse the core idea of some bufferlike processes of the modal model, we seek to draw attention to the need for a new class of models that can handle a range of working memory phenomena, not just the standard digit-span task.

In this article we describe one architecture from a class of architectures for working memory. We use the term "architecture" as it is used in computer science (see J. R. Anderson, 1983; Laird, Rosenbloom, & Newell, 1986), meaning a systematic approach to the configuration of computational components to accomplish some information-processing tasks. The proposed architecture illustrates both the limitations and capacities of human information processing. We also discuss human phenomena that identify qualitative features of human information processing and that should exhibit qualitative features of an architecture of working memory.

The connectionist/control architecture assumes processing occurs in a set of modules organized into levels and regions, e.g., vision, speech, semantic. The regions communicate with each other on an interloop of connections. This loop allows information to be transferred among input, output, and other regions, e.g., semantic or context. The information transfer within and among regions is modulated by a control processing system that controls the maintenance and output of information from modules. A new feature of this architecture is a proposed context-storage module that associates the content of messages in the interloop with the temporal context. The context storage system is able to reload modules after short-term information decays or is displaced. In addition, it provides a means of achieving stable, robust processing under conditions of high workload.

We define *working memory* in a manner similar to Baddeley (1986, p. 34) as "a system for the temporary holding and manipulation of information during the performance of a range of cognitive tasks such as comprehension, learning, and reasoning." To more temporally bound the range of working memory, we examine tasks in which the expected time to load an element into or retrieve it from working memory is less than a brief time (operationally defined as 10 sec). We are not overly concerned with categorizing something as long- or short-term memory, but rather, we define memory based on temporal dimensions and discuss experimental data and examples in terms of this new model.

We begin by reviewing the traditional models of short-term and working memory. We then describe a connectionist/control architecture for cognitive processing that describes the types of memory and processing strategies that exist in such a system. The new architecture relates three modeling themes. First, the connectionist structure draws heavily from the

concepts of connectionist modeling (Rumelhart & McClelland, 1986b). Second, the control structure is based on automatic and controlled processing theory (Shiffrin & Schneider, 1977; Schneider, 1985; Schneider & Mumme, 1987). And third, the combination of connectionist and control structures enables the architecture to accomplish many of the information processing operations associated with production systems (J. R. Anderson, 1983). We review a variety of literature on STM and provide an interpretation of it within the proposed architecture.

II. Traditional Views of Short-Term Memory

As noted above, two of the most influential models of STM were developed independently by Waugh and Norman (1965) and by Atkinson and Shiffrin (1968, 1971). Borrowing from James's (1890) terminology, Waugh and Norman proposed a model exhibiting independent primary and secondary memories. Primary memory was cast as a brief storage system markedly limited in capacity. This capacity can be roughly equated with a hypothetical buffer composed of a fixed number of slots. All information entering primary memory is either rehearsed or forgotten. If rehearsed, the information can be transferred to secondary memory from which it decays more slowly. Information can be lost from short-term store (STS) both as a function of delay over time and/or as a function of new items displacing old items. In other words, the longer an item resides in a slot without being rehearsed, the greater its degree of decay; an old item is thought be be displaced as a new item enters STS and occupies its slot. Note that in spite of the presumed rotational character of STS, early items from a list might not be lost if they are transferred into secondary memory.

Shortly after Waugh and Norman published their model, Atkinson and Shiffrin described mathematical models of learning and memory known as the Atkinson-Shiffrin model of memory (1968, 1971). Theirs too is a dual-component concept of memory, albeit one comprising a sensory register in addition to a STS and long-term store (LTS). This model was more differentiated than previous models, seeking to account for the richness of attention and memory phenomena; e.g., Atkinson and Shiffrin tried to specify how comparisons are made, how retrieval is controlled, and how items are transferred from the STS to the LTS. In doing so, they made the distinction between features of *processing structure* and *control processes*. The structure refers to the aforementioned register and stores, treated as a serial set of stages through which information is processed. The control processes refer to components of processing such as decision rules, organizational schemes, retrieval strategies, and problem-solving techniques. In contrast to the permanent structural components, control processes were characterized as optional, i.e., under the subject's direct control.

Baddeley and Hitch (1974) proposed a more complex STM system than those reflected in the unitary- or multiple-system theories of the late 1960s and early 1970s. They elaborated the idea of a working-memory system comprising separable subsystems. The articulatory loop is one of the subsystems, cast as a passive mechanism resembling a tape loop of limited duration used to store articulable information. In its later form (see, e.g., Baddeley, 1983, 1986), the articulatory subsystem is viewed as more active and made up of a phonological input store and an articulatory rehearsal process. A second subsystem is the visuo-spatial scratchpad, or as Baddeley (1986) prefers, the visuo-spatial sketchpad. This subsystem is described as being specialized to maintain and manipulate visuo-spatial images. It resembles the articulatory loop in that it is basically an input store. Further, it too is regarded as active in the sense that memory traces are thought to be regenerated by a process outside the store itself. Finally, the central executive is the subsystem assumed to coordinate information from the articulatory loop and visuo-spatial sketchpad. It serves the role of deploying attentional resources and of selecting and operating central control processes and strategies.

III. A Connectionist/Control Architecture for Working Memory

A. Architectural Principles

In this section we examine working memory from the perspective of a new architecture. Rather than using a traditional computer metaphor for the structure, we propose a commingling of ideas from neurophysiology, connectionist modeling, and controlled and automatic processing theory. Five principles suggest architectural constraints. First, we assume that processing occurs in a network of modules having a similar structure but differing in their inputs and outputs. This is suggested by the similarity of structure of hypercolumns of cells in the cortex of the brain (see Mountcastle, 1979), except that the hypercolumns differ in the input and output connections.

Second, we assume local specialization of function, i.e., that a given module specializes in a particular class of processing. For example, semantic modules may process words from a given semantic class. Evidence from neurophysiology suggests that a small region of cortex specializes in processing a small set of stimuli from a specific class, e.g., a 1 mm area of V4 visual cortex processes lines of given angles and colors from a 2° area of the visual field (Desimone, Schein, Moran, & Ungerleider, 1985). Cortical maps of the connection anatomy between regions of cortex are becoming very detailed in function (see Van Essen, 1985), suggesting there is a great deal of specialization of the connections among small areas, e.g., 10 mm² and localization of function.

Third, we assume that the knowledge is stored in the connection weights between neural-like units in the system. Physiologically the connection weights are likely to be the synaptic dendrite connections between neurons. The strength of the connection or the size of the weight is assumed to change with learning. The greater the weight between the input and output unit, the more the input unit activates the output. Storing information in connection weights is the defining characteristic of connectionist modeling (see Rumelhart & McClelland, 1986b; Schneider, 1987) and connections are very prevalent in the cortex. The connections provide an associative memory, such that a pattern in one module can evoke a pattern in another module. Associations are stored distributively, typically with many patterns in the same set of connections (see Hinton, McClelland, & Rumelhart, 1986). We assume that input to a module is a vector of activation, e.g., the letter *A* might be coded as 0, 1, 1, 1, 1 where the 0s and 1s represent the absence and presence of the features, e.g., vertical lines, horizontal lines, backward slant, and forward slant. The set of connections, i.e., association matrix, can store only one association per input vector, yet it can store approximately half as many random association pairs as there are connections. If input vectors are correlated, there is greater interference between the output associations (see below).

Fourth, we assume the connection weights may change with a variety of rate constants. The rate constants determine how rapidly the connections change as a function of interpolated learning and the retention interval. Hinton and Plaut (1987) have demonstrated that having fast and slow rate constants in connectionist models can speed learning, reduce retroactive interference, and speed recovery of previously learned material. They refer to connections with large rate constants as fast weights because they change quickly. Connections with low rate constants are called slow weights. At first glance one might object to multiple-speed weights as being unparsimonious; however, neurophysiological evidence currently points to the existence of over 50 neuromessengers, with time courses ranging from milliseconds to 30 min (Siggins & Gruol, 1986); even a simple motor ganglion synapse exhibits three time constants (Barrett & Mangleby, 1976). Consequently, it seems prudent to assume that multiple-speed weights exist, rather than a single weight.

Fifth, we assume a modulatory control system that regulates the flow of information among modules. This system has limited memory relating to control processes. It is the mechanism that produces attentional phenomena, in effect facilitating the sequencing and refreshing of information in the network. The control-processing system is a version of the system (CAP1) for implementing automatic and controlled processing (see Shiffrin & Schneider, 1977; Schneider 1985; Schneider & Mumme, 1987).

Mishkin, Malamut, and Bachevalier (1984) have proposed the existence of rapid and slow speed learning based monkey studies in which limbic lesions disrupt immediate memory for events occurring a few minutes prior without disrupting memory acquired slowly (i.e., after several trials) and tested after 24 hours.

We describe the architecture for working memory at three levels of detail. The microlevel represents a potential neural-like network that can produce associative processing and attentional phenomena, e.g., how visual features are associated to a code for a letter. The macrolevel represents the attentional control and message transmissions within the system, e.g., how memory scanning occurs. The system level represents the interactions among regions, e.g., how visual and auditory message transmissions are coordinated and how contextual biasing of message association occurs. The micro and macro levels of the model are the same as those used in the CAP1 model (see Schneider & Mumme, 1987). It is important to understand the relationship among the three levels. We recommend that the reader first get an overview of the three levels by examining Fig. 1-3, reading the captions, and then read the text. Readers who are more familiar with buffer models than connectionist models might benefit from examining the figures in a top-down order: the system level (Fig. 3), illustrating regional processors and levels of processing; the macrolevel (Fig. 2), illustrating buffer phenomena and sequential processing; and the microlevel (Fig. 1), illustrating how a neural-like system could store, categorize, and transmit information. The following text goes bottom up, i.e., micro, macro, and system, illustrating how each level is built from elements of the previous level of detail.

B. Microlevel Structure

Figure 1 illustrates the microlevel structure of the model. Information processing is assumed to occur in modules, e.g., M3 in Fig. 1. The *message* is represented by the state of the *output units* of the module. The set of activities of the output units is the *message vector* (MV) for that module, e.g., a code of 0, 1, 1, 0, 0, 1, 1. Each output unit sums the activity of its inputs. *Associative knowledge* is stored in the connections between the message vector and output units. Learning involves changing these connection weights. The activation of each output unit is a logistic function of its input. The logistic function produces a graded output as a function of the input, which has both minimum ~~which has both~~ (no firing) and maximum-firing levels of output. The output of the whole module is modulated by an *attenuation unit*. This unit modulates the vector messages as a whole. If the attenuation unit is fully activated, all of the output units are inhibited and no message vector is output from the module. If the attenuation unit is not activated, there is no inhibition and the output units transmit the message vector at full strength to the modules at the next level of processing. In the CAP1 simulation, attenuation is implemented by multipyling all the output units of the module by a fraction (the attenuation level) to determine the strength of the message vector. Within each module, different types of cells, called *report cells*, send scalar information to controlled processing. The activity report from the

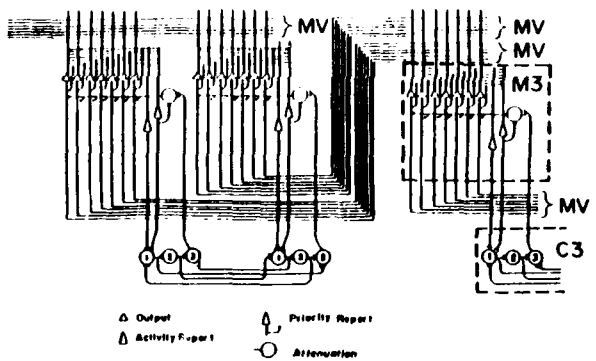

Fig. 1. Microlevel structure of the CAP1 simulation. Processing is assumed to occur in networks of neural-like units. Units are organized into modules (the box labeled M3 outlines the third module) that process a particular class of inputs. Information between modules is transferred as an MV on fibers connecting the output of one module to the input of the next. In the diagram, information flows from left to right (e.g., the top-left MV might encode visual features, the two left modules letters, and the right module words). Each module contains a vector of output units (the seven small triangles in each module). The output units receive input from other modules and connect autoassociatively to themselves. The recurrent connections from the bottom of each output unit going up and connecting to the other output units in the same module represent the autoassociative connections. Each of the crossing points above the output units (to message vector or autoassociative fibers) represents an associative connection whose strength of connection can be changed with learning. In the rest of the diagram the reverse arrow-type connections represent excitatory influences and the flat connections represent inhibitory influences. A module's output is controlled by an attenuation unit (the large circle) within the module. The attenuation unit regulates information flow from the module. Each module's activity is regulated by a control structure (the box labeled C3 represents the control structure for the third module). Each module reports its activity to the lower-level control structure via activity report and priority report units. The lower units (labeled 1, 2, 3) illustrate a potential control circuit. Cell 1 receives the activity reports from the module and inhibits the activity of neighboring modules. Cell 2 inhibits Cell 3, reducing the attenuation activation, thus reducing inhibition of the output units, thus enabling a message vector to transmit. Cell 2 is assumed to habituate, resulting in a burst of output and sequential switching of attention.

module to the control level communicates the module's assessment of the activity and importance of the current vector within the module.[1] The control level uses the activity report to determine whether an input is recognized, if there is a match between inputs, or that a module has something

[1] In the CAP1 simulation there are two types of report units. The *activity report* is a measure of how active the module is, e.g., the sum of the squared activity of all output units. The *priority report* is a within-module association between the vector message and a priority tag; it indicates how important the present message is for further processing. A local circuit allows the priority cell to automatically transmit the vector, i.e., without modulation by external control processes. This provides the mechanism for automatic processing which is the main topic of Schneider and Mumme (1987).

to transmit.[1] The control processing for the system provides a method to sequence message transmissions in the network. The microlevel structure is based on general features of cortical neurophysiology. The output cell units have connections similar to cortical-cortical pyramidal cells, the report cells to cortical-subcortical cells, and the attenuator cells to chandelier cells (see Schneider & Mumme, 1987; Szentagothai, 1979).

The box labeled M3 in Fig. 1 illustrates a simple circuit that allows messages from a set of modules to be output sequentially, as might occur in a memory-scanning experiment (Sternberg, 1966).[2] The control structure coordinates the activity of multiple modules. If multiple messages are transmitted concurrently, interference results and information is lost. The above circuit illustrates how the control system can enable one module's message transmission, while inhibiting neighboring modules' transmissions (see Schneider & Desimone, 1985, for details). The proposed microlevel structure shows some parallels with the available evidence of cortical hypercolumn anatomy (see Schneider & Mumme, 1987).

C. Macrolevel Structure

Figure 2 illustrates the macrolevel interactions of a set of modules. The macrolevel of the model has two types of processing—a message type and a control type. Message-type processing involves sending information messages, i.e., MVs of activation from one module to another. An MV would represent a large vector, e.g., of size 200 in the simulation. Control type processing involves monitoring the message traffic, clearing modules, and modulating the transmission of messages. These functions can be implemented with a circuit similar to that shown at the bottom of Fig. 1 (C3). Observe from Fig. 2 that there are three lines between the message vector and the control level. These lines carry scalar information and represent only a single fiber. The control region receives an activity report regarding the activity of the current message (see Figg1 and 2). The FEEDBACK signal sets the autoassociative feedback within the module (see Fig. 2). The *TRANSMIT signal (Fig. 2) enables the MV to be output by reducing the* amount of attenuation of the output units (see Fig. 1). The modules are arranged in levels and regions. Levels represent different processing stages,

[1] For details of monitoring activity reports using external control process modulation and the development of automatic processing, see Schneider and Mumme (1987).

[2] In the control circuit (see C3 in Fig. 1), memory is implemented by the state of habituation of Cell 2. If two modules need to transmit, they will both inhibit each other. The module with the higher activity/priority will win, blocking the transmission of its neighbors. While the higher activity message is being transmitted, Cell 2 will habituate (see Fig. 1 bottom). After habituation, the second module will transmit its message and inhibit the transmission of the first module's message, thus enabling a sequential readout of messages.

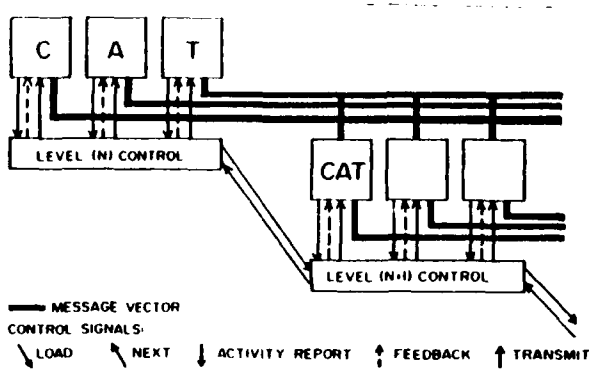

Fig. 2. Macrolevel structure. Each square represents a module in Fig. 1 (e.g., M3). The thick lines represent MVs that output an activity vector from one module to the next. The MV output flows from left to right. The thin lines represent control information between the level control and modules and between level control structures. The arrows indicate the direction of control information flow. The output of modules is modulated from a level control structure. (This is similar to the control circuit C3 of Fig. 1.) The control structure receives an ACTIVITY report from each module and outputs a FEEDBACK and TRANSMIT signal to the module. *The FEEDBACK signal determines the autoassociative feedback within the module.* The TRANSMIT signal reduces activation of the attenuation unit to allow output of the vector to other modules. Processing is assumed to occur in a series of levels. Each level communicates two control signals to the next level. The LOAD signal indicates a message should be loaded at the next level; the NEXT signal indicates the next level recognizes the message sent from the previous module and is ready for the next input. The figure illustrates how the sequentially loaded letters *C, A, T* can be transmitted as a group to the first word-level module of the word *CAT*.

e.g., visual dots, lines and bars, letters, and words, in which one level feeds information to the next. Each level communicates to the preceding and succeeding levels with two control signals. *The LOAD signal between .evel con*trol structures indicates that a level is transmitting to the succeeding level. The NEXT signal indicates to the preceding level that a signal has been *recognized and that the preceding level can reset itself in preparation for ad*ditional input from its predecessors. Regions represent sets of levels specializing in a particular type or mode of processing, e.g., visual, auditory, motor, semantic, and lexical. Modules at one level of processing transmit vector messages to the next level of processing.

The outputs from a region can occur in several modes. Modules can be loaded sequentially and then transmitted as a set to buffer input. Figure 2 illustrates how one might hear the letters *C, A, T,* sequentially buffer each

input, and transmit the set *CAT* to the next level of processing. To buffer output, the modules can be loaded as a set and then transmitted sequentially. The inhibition among the modules within a set (see C3 in Fig. 1) would produce the sequential behavior. This scheme is capable of implementing a sequential output system similar to the typing model proposed by Rumelhart and Norman (1982) (see also Section VII,A below). The sequential output mechanism can also be used to accomplish tasks such as memory comparison and visual search (see Schneider & Mumme, 1987).

D. System-Level Structure

Figure 3 illustrates the system-level interactions of regions of modules. Each region of processing may be a series of levels of modules and control structure as depicted in Fig. 2. Two types of processing that are analogous to the macrolevel exist at the system level (Fig. 3B). The central control structure receives activity reports from each region and modulates the transmission of messages in the central innerloop. The five control signals between the regions and the central control structure are analogous to those within a level, i.e., ACTIVITY, RESET, TRANSMIT, LOAD, and NEXT, except that the RESET signal involves resetting the control sequencing within a region, rather than changing the feedback within levels. The innerloop of processing refers to the communication between modules from each region that have connections to other regions. The central control system can be implemented using hardware similar to that used for controlling a single module (see C3, Fig. 1). The difference between the two is that the central control system receives the control signals from each region and routes the control signals among regions; e.g., if the motor region requests the next message, the central control structure may route the request to the speech-transmitting module. In contrast, within a region the LOAD and NEXT signals come from the next level within a region.

The central control structure modulates the output of regions transmitting on the innerloop. As a result of preprocessing in each region, the central control structure need only process a single scalar value (the activity report) from each region and not directly deal with vector messages. The distribution of control among modules, regions, and the central control system avoids the homunculus problem, i.e. delaying all complex processing to a later stage that cannot be specified.

An important feature of the system level is that there is no central executive through which the messages pass. A simplified system might have a single region receiving input and output from all other regions (see Baddeley, 1986; Barnard, 1985; Broadbent, 1984). The advantage of the current cross-connection system is that each region can communicate with other regions without passing through another module. This enables faster single-message transmission and allows multiple regions to jointly activate a

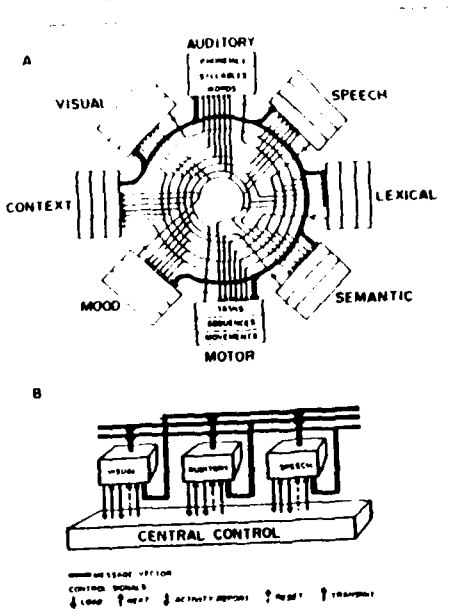

Fig. 3. A. System-level description of the model, a top-down view of the regions of processing within the system. Each region represents a series of processing levels, as in Fig. 2. The first or last level of a region (last level for input regions and first for output regions) is assumed to input to the innerloop of connections between regions. The modules on the innerloop have separate message vectors to each of the other modules they connect to. All the lines in Fig. 3A represent MVs (see Fig. 1). The context module sends a MV to all the other modules on the innerloop. The output for the context module is highlighted to illustrate this connection pattern. This figure represents a simple view of one of many possible connection patterns for regions on the innerloop. B. Side view of the system-level architecture. All of the regions of the innerloop connect to a central control system that routes control signals between regions on the innerloop. The system manages message traffic on the innerloop to maintain reliable communications across regions.

region. If regions automatically transmit their high-priority messages (see Schneider & Mumme, 1987), small numbers of high-priority messages can be rapidly processed without requiring controlled processing. The disadvantage of a cross-connection system is that concurrent message transmissions can cause interference. To avoid such interference, the central control structure sequences the transmissions serially. This forced sequencing can result in delay or omission of messages; e.g., if too many messages

(see Schneider & Detweiler, 1987 for discussion of compensatory activities to limit interference).

wait for transmission, the regional activity report may decay before the central controller enables the transmission.

The message vector connections among modules are not a single pathway or bus. Each region has its own set of fibers to other regions. In Fig. 3A, the darkened line from context illustrates how each module has its own pathway to the other modules. Interference is not based on whether a vector is being transmitted, but rather on whether the receiving module receives competing messages. Note that there is an independent association matrix between each input and output module, i.e., a set of connections between the transmitting vector and each receiving module. Thus the auditory region might transmit a vector for the sound of the word *pen* to the semantic region (evoking the meaning of the object) and to the speech region (evoking the speech output of the word). At the same time, the visual system might transmit a vector representing the visual features of a left arrow (evoking a left motor movement in the motor region), and a change in orientation in the spatial region. A neural system with large vectors could support many dissimilar concurrent transmissions without substantial degradation of performance.[4] This system contrasts with Broadbent's (1984) Maltese Cross view of memory in which all messages must flow through a single central-processing system. It also contrasts with single-bus architectures (such as Barnard, 1985). It is not the number of messages transmitted, but rather the number of competing messages received that determines limits on the number of concurrent message transmissions. If the region can determine when it is receiving interfering messages and signal the central controller, the central controller can then begin sequencing transmissions of regions waiting to transmit.

E. Context Storage

Context plays a critical role in maintaining working memory in a system with many connections. For purposes of illustration we have labeled one of the regions in the innerloop as context (see Fig. 3 left side). We assume that context is a continuously varying representation of the internal state of the individual. It could be implemented in several ways. For instance, context could be the current state of the system, e.g., time of day, emotional state, hunger, or it could be a randomly varying vector, e.g., with each unit of the vector having a probability .5 of changing state every minute. The context vector can output its message to the innerloop. As with the other regions, an association matrix exists between the context vector and each region to which the context vector connects. Within a region, the context vector may connect to many modules. The context associations provide the potential

[4]An order of magnitude of 10,000 would seem reasonable based on the physiology of hypercolumn interactions (Mountcastle, 1979).

The proposed system-level architecture parallels the amygdaloid complex in the brain, which is characterized as having direct and extensive connections to all cortical sensory systems (see Mishkin & Appenzeller, 1987). The convergence and interconnections of sensory regions to the amygdala could provide an innerloop style of communication.

for a very large working memory storage. To illustrate, assume one has R regions and N modules connecting to the innerloop from each region. The context vector could then be associated to $(R - 1)N$ modules. Assume further that there are 20 regions in the innerloop and 30 modules in each region that connect directly to the innerloop. A single context vector and its association matrices could store 570 codes (one/module). It would be unlikely that all the regions would ever have an association to a specific context code (see below). However, the system would certainly have the potential for storing much more then the 5 ± 2 typically associated with STM (Mandler, 1967).

Storage is dependent on where learning takes place and what code is active in the receiving module. In the CAP1 model, associative learning occurs whenever a vector input is followed by a TRANSMIT-activated release of the vector that was previously stored in the module.[5] Using the same learning rule here, the only connections modified after a transmission are those that were activated by the input and were connected to a module that received a TRANSMIT control signal after the input transmission. For example, assume a transmission occurs from a module in the auditory region and the context region, followed by a controlled-processing release of a vector from the motor-system module that controls the hand. The only connections that would be modified would be those on incoming auditory and context fibers in the motor module controlling the hand. Although many modules may receive the auditory transmission (see Fig. 3A), only connections in the transmitting modules are changed.

Attention in this architecture is the TRANSMIT-activated output of the MV (see Schneider & Mumme, 1987). Attention allocation is limited and varies with subject strategies. What is actually associated to the context is limited by what is attended (see Fisk & Schneider, 1984).

The availability of stored information is dependent on the learning constant and decay rate for connection weights. Most connectionist models use some type of delta learning rule in which the connection strengths are changed as some proportion of the difference between the output vector and the input vector (see Hinton & Sejnowski, 1986). The proportion ranges from 0 to 1.0, with values less than .1 being typical. The larger the learning constant, the smaller the number of learning trials needed. However, the larger the learning constant, the more serious is the problem of retroactive interference (see below).

[5]If CAP1 were to allow associative learning to occur after an automatic transmission, the association matrices would deteriorate and the network could no longer perform comparison tasks (Schneider & Mumme, 1987). Limiting learning to only modules that are attended, i.e., that receive TRANSMIT signals, coincides with experimental data suggesting that learning occurs only after controlled processing and not after automatic processing (see Fisk & Schneider, 1984).

The decay rate of connection weights determines how long the previous associations influence the output of a vector. *Fast decay weights are advantageous* because the association matrices can be used to provide a working memory that is unaffected by the information stored, say, five time constants earlier. For example, if the weight were to decay to half strength in one minute, there would be no proactive interference from any vectors stored five time constants earlier—the connections would have decayed to only .03 of their original strength. Fast decay weights are disadvantageous because the decayed information can no longer be retrieved, e.g., an *association learned five constants earlier cannot be retrieved.*

A system with multiple learning and weight decay rates can substantially enhance the performance of the network. Hinton and Plaut (1987) have found that having both fast and slow weights can substantially speed learning. Typically one set of weights has a high learning rate and decays rapidly. A second set of weights has a lower learning rate and decays more slowly. The fast weights learn quickly so that after a small number of trials the input can evoke the output. During the same trials, the slow weights change gradually. After a large number of trials the fast weights have less influence due to the effects of retroactive interference, weight decay, and the buildup of the slow weights (see Hinton & Plaut, 1987, and below). This multiple-weight scheme allows the network to temporarily alter the connectic weight space so that older memories can be recovered. For example, assume someone has learned a foreign language as a child but does not speak it regularly. The connection weights between the semantic and speech modules are modified as a result of practice with the current language. In a situation in which use of the first first language is necessary, the fast weights change during the first few minutes of conversation. This alters the connection space so that it more closely approximates that of the first language. *Note that although much of the previous first-language knowledge returns,* no significant change occurs in the long-term weights. As the person resumes use of the current language, the short-term weight changes may temporarily reduce availability of the current language. However, after a change in context or waiting five time constants, the person should be able to operate in the current language with no deficit.

For the present discussion of working memory we assume that the context-region connections have fast learning and fast decay rates compared to the rates of the other regions in the system. It may be the case that all of the connections in the system have both fast and slow change rates. Nevertheless, for purposes of illustration it is easier to refer to (1) the context weights, implying that these are the fast learning rate and fast decay connections, and (2) the information weights (connections other than context weights), implying that they have relatively slow learning and decay rates. To help illustrate these ideas, consider a simple learning example. If

This separation matches physical separation of rapid and slow learning (Mishkin, et al., 1984) occurring in different parts of the brain.

one were to try to associate a visual shape to a word presented auditorily, a fast-weight change would occur between the context and the speech output region, and a slow-weight change would occur between the visual and speech systems. After learning a single paired associate, one could perform 20 reaction-time tasks, e.g., saying the word for the shape using primarily the context associations. However, if one had to remember the shape-word pairing *after an hour or had to learn five sets of such pairs, the context* associations would be of little value due to the effects of weight decay or proactive interference with the other learned codes.

The issue of working memory in this architecture is multifaceted, consisting of many memories within the system. At the microlevel, the activity level of each output unit decays with some time constant. Each module is assumed to have feedback connections that result in the categorization and maintenance of codes. The combination of feedback and decay may allow a module to maintain a well-known code indefinitely,[6] assuming there are no new inputs to the module. In order for information inside a module to influence the rest of the network, it must be transmitted out of the module. *This depends on the activities of the regional controller. This controller is* assumed to have memory concerning which modules have been transmitted and of the activity and priority of the messages waiting to transmit. The regional controller may function as a buffer memory allowing the storage of a few vectors. In this form the regional controller illustrates phenomena similar to the buffer models of STM, e.g., Atkinson & Shiffrin (1968). In addition to the memory resulting from dynamic activity, a great deal of knowledge can be stored in the connection weights in the network. Much of this probably represents slow weights and should be considered LTM. However, we assume there is at least one set of fast weights connected to a context vector that is broadly connected to at least the innerloop of processing regions. This vector facilitates one form of intermediate storage and enables context storage of information. By associating vectors to context, working memory can be reloaded by transmitting the context vector.

F. Simulation Methods

The simulation results described in this article were run using the CAP1 simulation program described in detail in Schneider and Mumme (1987). The model includes the associative and autoassociative models of J. A. Anderson's (1983) brain-state-in-a-box model. The model is a connectionist model with a control structure. The results in this article represent robust characteristics of the architecture and are not dependent on detailed parameter searches.

[6] The feedback only helps for patterns the module has previously learned (see Schneider & Mumme, 1987).

The components of the model are illustrated in Fig. 1. Each module is made up of a 200-element vector of output units. Each output unit sums its input linearly with the decayed value (typically 0.9) of the activity of the unit on the previous iteration. The output of a unit is a logistic function of the input with limits of + 1.3 and - 1.3 activity. Each output unit connects autoassociatively to half the other output units in the module. Autoassociation provides feedback; the strength of feedback varies between 0 and .6, depending on the level of control feedback input (see Fig. 2) to the module.

Each output unit connects to half of the other output units in the same module and to half of the units in other modules. With 200-element vectors, a single output unit connects to 100 output units in its own module and 100 units in every other module that receives the message vector. The autoassociative and associative connection matrices are each 20,000 connections per module.[1] The associative matrices between modules were initiated with a zero strength of connection between all elements. Learning was accomplished using a Hebb-type learning rule that modified the strength of connections so the input pattern would come to evoke the output pattern. CAP1 uses a delta or Widrow-Hoff learning rule (see J. A. Anderson, 1983; Kohonen, 1984). The strength of connection between the input and output is updated by a learning constant multiplied by the differences between the desired output and the output evoked by the input. In terms of matrix algebra, the change rule was *delta* $A = C(R - AS)ST$, where A is the associative matrix, R the response (desired output) vector, S the stimulus vector, ST the transposed stimulus vector, C the learning constant, and *delta* A the change in the strength of association.

The output of the module is determined by the attenuation unit. To transmit a message from a module, the level control would activate the TRANSMIT signal (see Fig. 2). This signal inhibits the attenuation unit (Fig. 1), allowing the output units to transmit the message vector to the modules at the next level of processing. The attenuation is a multiplication of the activity of the output units (when transmitting, the strength is 0.3 or 70% attenuation; when it is not, it is 0 or 100% attenuation).

All input codes are random 200-element binary vectors with a specified correlation. All the output codes had a zero correlation. The sequential context vectors were correlated to the previous context vector (typically .9). For example, between trials 10% (20 elements) of the context was changed from trial to trial.

Before a simulated experiment was begun, the autoassociative matrices within the modules were taught the ensemble of target patterns. This is

[1]The results described here generalize to much larger matrices. The physiological data suggest a much larger number of connections are involved even in small regions of cortex (Mountcastle, 1979). Simulations with small matrices do not generalize due to the effects of spurious correlations with small numbers of elements.

analogous to testing subjects in a STM experiment and assuming they enter the room with knowledge of English and will be tested using high-frequency words. Typically the system was taught 50 random vectors presented 10 times each. This modified the autoassociative matrix on each presentation with a decaying learning constant of .1 on pass 1, .09 on pass 2, .081 on pass 3, to .039 on pass 10. After this procedure, the autoassociative matrices would produce positive feedback that matched (correlation above .99) the input for all the patterns (see J. A. Anderson, 1983; Schneider & Mumme, 1987).

When a module transmitted to another module, it transmitted a short burst, typically eight iterations of output to the next module.[1] This produced a short burst of output to the next stage. The next module performed autoassociative processing during and after the burst to receive and clean up the vector message.

The dependent measure of memory retrieval for the simulations is the percentage of vector match between the retrieved vector and the desired vector. An output unit could be in one of three states: high, activity above .5; neutral, activity between .5 and - .5; or low, activity below .5. The degree of vector match was based on how many elements matched between the retrieved and desired vector using a three-state city-block metric. By chance, vectors should have an average error of one/output unit. The percent of vector match metric is the percentage above chance that the retrieved vector matched the desired vector. A 100% match is a perfect match, a 50% match implies an exact match on half the vector and chance match on the rest, a 0% match implies only a chance match. The probability of recalling an item would be a monotonic function of the percentage of vector match. The actual recall rate would depend on the number of vectors learned, feedback, and similarity of vectors. The reader should treat the percentage of match as a simple approximation of the expected recall.

All simulated recall trials were run until the network settled on a vector representation. This typically required fewer than 10 iterations. Each iteration involved four components. First, the activity of the output vector was decreased by the decay rate, e.g., each output unit's activity was set to .9 of what it was on the last iteration. Second, the input vector was multiplied by the associative connection matrix (see the cross connections between message vector and output units in Fig. 1). This produced the input activity vector (one element of the vector for each output unit). This vector was multiplied by the feedforward (or associative) constant and added to the vector of output units. This occurred for the first five iterations after the stimulus was presented. Third, the output vector of the previous iteration

[1]This bursting can be accomplished by having the control cells habituate. For example, if Control Cell 2 in Fig. 1 habituates, the system will tend to output in bursts.

was multiplied by the autoassociative connection matrix (see M3, the cross connection between the output units to themselves with the module in Fig. 1). This was multiplied by the feedback (or autoassociative) constant and added to the input of the output units. Fourth, the activity of each output unit was set to a logistic function (a sigmoidal transformation) of the input to the unit. In memory, the input for each unit at Level 2 would be

$$I_{2i} = dO_{2i} + f\sum_{k} O_{1k}W_{ki} + b\sum_{j} O_{2j}W_{ji} \quad (1)$$

$$O_{2i} = -m + \frac{2m}{1 + \exp(-aI_{2i}/m)} \quad (2)$$

where the first subscript represents the level of processing, the second unit within the level. The constants are d, decay; f, feedforward associative input; m, the maximum absolute value of the activity level; a, the slope of the logistic function; and b, feedback autoassociative input. I is the input activity; O the output activity, and W the connection strength.

In the following discussion we describe only changes from the above parameters. The default parameters were the following: $B = .6$; feedforward to the next stage, $f = .3$; decay, $d = .9$; slope of logistic function, $a = 2$; number of iterations of a burst, 5; correlation of context vectors, .9; correlation of response vectors, .0; number of iterations before match determination, 8; and learning constant, 0.1. Typically 4 to 24 simulation runs were sufficient to produce stable data.

A typical free recall sequence involves presenting a list of vectors to be learned and then having the network recall the vectors. On each trial the connection weights between the input and output vector are modified using the delta rule. The matrix is then presented with the input, whereupon the output is evoked and categorized via autoassociative feedback. The percentage of vector match is calculated, and this is considered the measure of learning. At the end of learning a series of vector pairs, all of the inputs are represented to the matrix and the end-of-list vector match is calculated. The end-of-list percentage of vector match is a metric for retention.

IV. Interpretation of the Working-Memory Literature

The present architecture provides an approach for representing a class of models of working memory. Within this architecture the modeling process is one of weaving together a set of parameters, modules, and connections to produce specific models for particular phenomena of working memory. We hope that a very similar structure can be used to represent many regions of

processing. However, the brain is not a parsimonious processing system. We expect the human processing architecture to include many variants of one, or perhaps several, architectures. In this section we describe how the major phenomena of working memory can be interpreted within the connectionist/control architecture. Illustrative simulations of some of the core concepts are presented. The full elaboration of this system will require extensive research and modeling which we hope will be accomplished by a variety of researchers.[9]

The architecture includes a variety of processing elements to accomplish stable processing of real-world input. Context-based storage is critical to prohibit catastrophic failures of memory. For example, if one is distracted by a phone call in the midst of writing, one can reengage the ideas focused on when interrupted, even though the phone call may have required temporarily storing 20 chunks of information. The control of module transmissions is necessary to allow one to attend to information in situations with multiple stimuli, or information overload (see Schneider & Mumme, 1987). The sequential loading and transfer from regions is necessary to handle sequential input or output. The use of the central controller is necessary to allow multiple regions to evoke common codes while limiting message interference so that critical messages are not blocked, e.g., either seeing a red light or hearing the word *brake* can cause one's foot to press the brake pedal, but one can also selectively ignore, or at least reduce the influence of, one of the modalities. We now examine how particular features of the architecture can be used to interpret working memory phenomena.

A. Buffer Phenomena

As mentioned at the outset of this article, the classic or "modal model" (Baddeley, 1986) of STM is a buffer model of the type exemplified by the Atkinson and Shiffrin (1968) model. Three major phenomena are often cited in support of buffer models. These phenomena are very stable and have probably been replicated hundreds of times (see Baddeley, 1976, 1986, for reviews). The first is the *recency effect* in free recall. When subjects are given a list of words to recall, the last few items are recalled dramatically better than the rest of the items (see Postman & Phillips, 1965). Items in the buffer are lost due to displacement rather than due to time delay. For example, Baddeley and Hitch (1977) had subjects classify a list of 12 names as male or female followed by either immediate free recall, a 30-sec blank delay, or 30 sec of copying digits. In the first two conditions, recall of the last item was over 90%, whereas in the filled delay it was only 52%. The

[9] We are developing computer modeling tools we can provide to other researchers to help expedite explorations of and extensions to this architecture.

lack of a delay effect in the blank interval indicates that trace decay was not a major factor in producing the recency effect. A second related phenomenon is the STM *decay effect* caused by interference. Peterson and Peterson (1959) showed that the probability of recall dropped from 80% to 10% in an exponentially decreasing function if subjects counted backward by 3s for 3 to 18 sec respectively. A third related phenomenon is a *span effect* based on studies of digit and word span. When presented a string of digits, letters, or words, subjects can typically recall an average 8.2 digits, 7.2 letters, and 6.3 words, when span is defined by the length of list recalled correctly 50% of the time (Crannell & Parrish, 1957).

The present architecture includes buffers in the regional controllers and potentially at multiple levels within the regional controllers (see Fig. 4). Many such buffers exist, and they are essential for the sequential input and output of information. For example, in speech perception typically one to three phonemes make up a syllable, one to four syllables make up a word, one to four words make up a phrase, and two to four phrases make up a sentence. The storage capacity for a specific task depends on the specific codes, stages, and regions involved. If the higher-level representations have codes for the lower-level sets of buffers, a region may be able to store as many words as

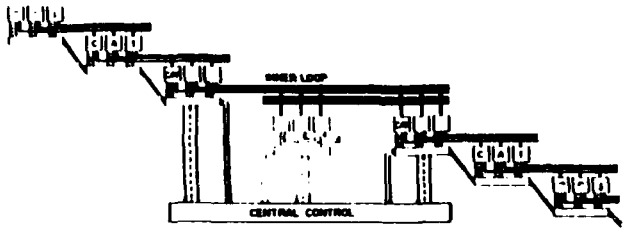

Fig. 4. Levels of processing from input to output. This diagram represents the activity of patterns in the network that would occur after sequentially reading the letters for the word *cat* and outputting the motor movement to reproduce the word. The codes in the boxes show the state at the time of the last motor movement. At the top level, the features of the last letter are active. At the second level, visual feature sets code the letter positions. At the third level, the three letters have been combined into a single visual word code. This visual word code is transmitted as a single visual message to all the modules on the innerloop. The Level 4 motor unit receives the visual code *cat* and translates it into a motor code for the task of reproducing the word *cat*. The motor-task code is transmitted to the motor-sequence modules which store each set of movements for each letter. The "T" motor code is transmitted to the motor movement modules converting the code into three separate movements. The motor movement modules are sequentially output, causing the sequential strokes to produce the letter *T*. The dotted boxes in the center of the diagram represent other regions on the innerloop not involved in this specific message transmission. The context storage mechanism could enable reloading of the modules on the innerloop. The control signals are the same as those described in Fig. 2 and 3.

consonants, because there may be as many modules for words as there are modules for consonants. Human STM is essentially equivalent whether it is measured in number of consonants or words (see Murdock, 1961).

To implement a buffer scheme in the proposed architecture there must be methods to clear a module, maintain a code within a module, output the code from a module. Clearing a module can occur by several methods, including (1) inhibiting all the units to zero activity, or (2) increasing the decay rate and reducing the feedback. To clear memory in the simulations we used a decay time of 0.07 in the absence of feedback, i.e., with feedback set to zero. This effectively clears the memory for a module in about 0.2 sec.

Maintaining a code in a module is accomplished by a combination of activation decay and feedback within the module. Within each module an autoassociative matrix (see J. A. Anderson, 1983) associates each previously learned vector to itself. This has the benefit of cleaning up and categorizing noisy inputs (see Schneider & Mumme, 1987; Anderson & Mozer, 1981). With a feedback parameter of .4 and a decay of .9, the system can maintain a well-learned code indefinitely. Note, however, due to the critical role of feedback the module cannot maintain novel codes unless they are similar to previously learned codes; this is because the net feedback is zero for novel codes dissimilar to the learned codes.

In order to buffer and hierarchically code information, all of the modules at one level of processing must feed into each of the modules at the next level of processing (see Fig. 4). If the input "C," "A," "T" enters sequentially while reading, it is critical that the module containing the first letter not be displaced by the second or third letters. One method to implement this involves multiplexing the input from the previous stage. If the feature level could selectively gate input to each letter module, loading the second letter would not interfere with the code in the first letter module. However, this requires substantial hardware at the front end of each stage of processing and is unnecessary. Feedback locking to lock codes in buffers so that the input to a second module at a level will not interfere with the code contained in the first module. If a code is loaded into a module and maintained by high feedback, the combination of the feedback and the nonlinearities of the output function can block the interfering effects of input codes that are dissimilar to the loaded codes. In the simulation, increasing feedback to .6 after the module is loaded will maintain the code within a few percent, even if other dissimilar input codes are activated to load neighboring modules. To illustrate, we loaded vectors A, B, C into three modules, each of which could store any of the vectors (see also Fig. 5). The accuracy of storing the first vector in Module 1 was .964 after A was loaded into the first vector, .947 after storing B into Module 2, and .943 after storing C into Module 3. An easy method to eliminate the effects of loading a vector on the to-be-loaded modules involves clearing the module in the process of loading the current module. This is the scheme we

To load a new vector into a module, we set feedback to zero during the initial load process. This clears the old message.

use in the simulation. When a level receives a LOAD signal it sets the feedback of the modules storing the information to zero. This causes the old vector to decay and the new vector to be activated; the feedback is then increased, locking the code in the module.

The proposed sequential loading scheme has trouble loading similar codes *into neighboring modules, thus predicting human problems with* similar input. The reason for this is that the feedback effect can easily block an orthogonal code but it has difficulty blocking a similar code. Sequentially storing two similar vectors (e.g., *with half the elements equal*) into neighboring modules at the same level can cause several times the error of storing random vectors. The size of the disruption depends on factors such as the degree of similarity of the input and output codes, the number of codes, and the duration and strength of feedback.[10] Storing similar codes disrupts the noncommon portions of the code. This increases the probability of confusions but generally does not result in omissions.

To appreciate the influence of code similarity on processing, recall that Conrad (1964) has shown that most errors in STM are acoustic confusion errors, even when the information is presented visually. This would be expected if the vector code for the information were acoustic and should be more likely if the other items in the remainder of the list were acoustically similar. Baddeley (1966) illuminated this effect further by conducting an experiment to assess whether STM would be more disrupted by acoustic or *semantic similarity. Here subjects were presented sequences of five* acoustically or five semantically similar or dissimilar words and asked to recall each sequence immediately following its presentation. The influence of acoustic similarity was striking; subjects correctly recalled only 9.6% of the similar sequences as opposed to 82.1% of the dissimilar ones. The influence of semantic similarity was much smaller; in this case subjects correctly recalled 64.7% of the similar sequences and 71.0% of the dissimilar ones.

The last step of sequential processing involves having all active modules output as a set to the next stage of processing. For example, after the elements "C," "A," and "T" are sequentially loaded into the letter level, the entire set can be transmitted in parallel to activate the word *cat* at the next level. This can be accomplished by reducing the activation of the attenuation units of all the active modules in a level (see Fig. 1). As long as there are few interconnections among modues at the same level, transmitting information out of a module will not disturb the contents of the other modules as the same level. This is important because it allows one level to

[10]We are in the process of investigating these relationships. Currently, presenting the module-related codes reduces the likelihood of the module retaining the noncommon portions of the code, resulting in more confusions.

output to the next level with every new input until the next level recognizes the set of inputs. In this way higher-level modules can detect matches even without a blank. For example, for the input "C," "A," "T" the outputs would be "C," "CA," and "CAT," at which point the next level would attempt to recognize the input via autoassociation (see J.A. Anderson, 1983; *Schneider & Mumme, 1987) After recognition at level N + 1, that level* would return a NEXT signal (see Fig. 2) to reset level *N* so the next set of elements could be buffered at level *N*.

Sequential information can be encoded and stored in several ways. The first scheme involves positional coding. If the modules at a level are filled in a prescribed order, the connections for each module can provide positional markers, i.e., we assume that each module has separate fibers to the next level, and that each set of fibers codes positional information. The disadvantage of this scheme is that the level must always know which position it is in, and the later modules in a level are rarely used.

A second scheme for sequential storage is to have each module use a context-sensitive code to code its information (see Wickelgren, 1969). Rumelhart and McClelland (1986a) have used such a scheme to code Wickelfeatures (Wickelgren, 1969) to encode speech sounds. For context-sensitive coding, each module must include in its code the present input plus at least some features of the previous and following input items. For example, to code the input "C," "A," "T" in a context-sensitive code, each module would code "-Ca" "cAt" and "aT-" (where the - indicates a terminator and the lower-case letters degraded or coarse coding of the neighboring letter). With context-sensitive coding the letters can be *represented in any module and the next level would detect the word cat.*

The advantage of a context-sensitive coding scheme is that the inputs can be in any position as long as the same context-sensitive code does not reappear in the input. In speech production, such a coding scheme allows a position-independent representation of nearly all the words in spoken English and can predict many speech errors (Wickelgren, 1969). For the purposes of the present model, such a code could allow sequential storage without requiring specific sequential positions to always occur in specific modules. Thus "CAT" could be represented in many spatial positions, or perhaps in an ambiguous context that could not otherwise be interpreted, e.g., "XCAT."

A third scheme for sequential storage is dynamic reallocation. In this scheme all of the memory at one level of processing is allocated for storage of the first input and the modules are reallocated to store additional information as needed. For example, assume the sequential string ABCD is stored in four modules. Input A would be stored in all the modules, AAAA; B in one-half, AABB; C in one-fourth, ACBB; and D in another one-fourth, ACBD. Position information could be encoded either by position

markers, e.g., with Module 2 encoding position 1, 1, 3, 3 if there were 1, 2, 3, or 4 codes stored in the level, or by a context-sensitive code. The advantages of dynamic reallocation are the following: (1) all of the memory is always used, thus producing more reliable and faster processing for smaller numbers of items; (2) the system need not know the sequence length at the beginning of the sequence; and (3) the system degrades gradually with list length, rather than processing, without an error and then always failing, as in a fixed-slot buffer memory.

With dynamic reallocation or context-sensitive coding, the modules at one level of processing can be treated as a ring buffer, rather then as a set of fixed positional slots. In a ring buffer the first element of the ring is the position next to the last, and given that it is a ring, there is always a next element to store into. After all of the slots have been stored once, the next item stored replaces the oldest code in the ring. In a ring buffer of size M, one can store the last M sequential inputs. For example, assume the buffer has seven slots. As the sequential input "CATCH" is entered, the codes "-CAT" and "-CATCH-" could be identified. By varying the number of elements back that are activated, groups of one to seven letters could be output to the next stage of processing. Having information stored in a ring allows higher levels of processing to review the previous input until the limits of the ring are exceeded. Baddeley (1986, p. 80) suggests that the articulatory loop can store the last 1.67 seconds of reading time for short words. Assuming it is always the last 1.67 seconds, this suggests a ring-buffer type of allocation, rather than fixed slots which are reset after each punctuation mark.

The process of sequencing items serially followed by a parallel output to the next level is illustrated in Fig. 5. The activation of each code and the changes in the control parameters are shown. The simulation incorporates feedback locking in order to lock codes in modules. Observe that the feedback parameter is used to clear modules (by being set to zero) and to maintain the modules' codes while neighboring modules are loaded. Dynamic reallocation is used, i.e., modules are cleared, when needed. The code "CAT" is stored as "CCC," "CAC," "CAT" for each input. The LOAD signal from the letter-feature level (line 1, iterations 1–8) causes the letter-level controller to deactivate feedback (lines 3, 6, 9, iterations 1–8), loading "C" into all the modules. In iteration 9, the letter-level LOAD signal is deactivated, activating feedback (lines 3, 6, 9), and increasing the activity in Modules 1, 2, and 3. The rapid activity increase (iterations 9–12) indicates to the level controller that the code has been recognized. The letter-level controller sends a NEXT signal (not shown) to the letter-feature level, causing the letter-feature level to be cleared. The letter-level controller sends TRANSMIT signals (lines 4, 7, 10, iterations 10–16) to the letter modules, outputting the "C" code to the word level. It also sends a LOAD signal to the word-level controller.

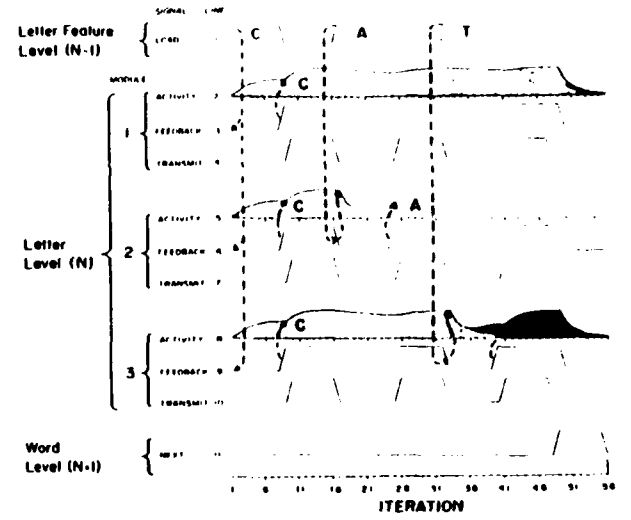

Fig. 5. Sequential input dynamics. This diagram shows the activity levels and control signals for sequentially loading three vectors in a CAPI simulation. This represents the activity of levels 1, 2, and 3 of Fig. 4. The ACTIVITY signals (lines 2, 5, 8) show the activity of all of the output units of the vector. The maximum points on the line represent an absolute value of 1.1 units of activation. The FEEDBACK signals (lines 3, 6, 9) control the autoassociative feedback within the module. The high state represents a feedback constant of .6, the low state a feedback constant of 0. The LOAD signal (line 1) indicates the letter-feature modules transmitting a control signal to the letter-level modules (lines 2, 5, 8). This causes the letter level to reduce the FEEDBACK (lines 3, 6, 9) so the letter-level modules accept the transmitted message. The vertical dashed lines represent a causal influence in the diagram. The downward lines represent the LOAD signal altering feedback. The upward lines represent feedback changing the activity of the appropriate module. The TRANSMIT signal (lines 4, 7, 10) represents the outputting of the message vector to the next module in the sequence. At the same time that the TRANSMIT signal is activated, a LOAD signal is transmitted to the next level (not shown). As the next level recognizes an input pattern (for the pattern "CAT"), it returns a NEXT signal to the letter level causing the letter level to clear its memory and send a NEXT signal to its predecessor level.

On iteration 16 the LOAD signal (line 1) is activated for the second set of letter features. The letter-level controller dynamically reallocates a portion of the modules to store the incoming information and reduces the feedback of a module (line 6, iteration 17). With no feedback and new input, the old code is replaced with the new code ("A" replaces "C" in Module 2).

Notice that the transmission of the "A" code did not significantly influence the activity of modules for which feedback was still activated. The slight decrease in activity (lines 2 and 8, iterations 17–24) shows the minor impact of the modules receiving the "A" code while the "C" code is maintained by the feedback within the module. All the modules transmit their contents ("CAC") on iterations 26–32. The third letter is allocated to Module 3 and replaces the "C" code. On iteration 47 the word-level module recognizes the code and returns the NEXT signal, causing the letter-level to reset."

The dynamics illustrated in Fig. 5 show how the architecture in Fig. 1 and 2 performs sequential input processing. Information is buffered at each level producing asynchronous input processing. Each level can hold or retransmit its information if the next level does not respond. The next level can detect patterns even without explicit boundary terminators, e.g., in the previous example, "CAT" was recognized because it was the first meaningful code encountered in the pattern "C," "CA," "CAT," not because there was a blank afterwards.

The immediacy of perception in reading (see Just & Carpenter, 1987) is consonant with the proposed processing dynamics of the model. Just and Carpenter assume that at every level of processing there is a chunking process that stores the sequential inputs of the previous level and tends to clear the previous level. The lower the level of processing, the more frequently the buffers are cleared. For example, with each eye fixation the visual features of the previous fixation seem to be lost. McConkie and Zola (1979) examined students reading of AlTeRnAtInG cAsE text. During some saccades the text of every letter was changed (e.g., cAsE becomes CaSe). This change was not perceived and had no effect on eye movements. The visual features relating the type which specify the difference between the upper- and lowercase letters are not integrated across fixations in reading. Jarvella (1971) found analogous results for the processing of clauses. He found people tend to forget the exact wording of a preceding text more rapidly if it is in a previous sentence than in the current sentence, even when the same wording is present. For example, recall of a six-word phrase was 42% if it was at the end of the previous sentence and 82% if it was at the beginning of the current sentence. The break was quite discontinuous, affecting all words in the clause. These data are consistent with a view that sequential processing involves a multilevel sequence of buffers where the higher-level buffer stores a representation of multiple items from the previous level and storage at the higher level tends to clear the previous level.

"The NEXT signal could automatically cause the level to prepare to clear all modules with the next LOAD signal from a preceding input. By delaying the act of clearing information, it would be possible for the level to retransmit information if needed.

B. Multiple Buffers

The number of buffers available at a level of processing may vary depending on the nature of the material. Some buffers may be organized sequentially as in audition, some spatially as in vision, or some based on content as in semantic memory. As a single level is dynamically reallocated for greater numbers of buffers, the quality of the coding degrades. We assume there is a limit to the number of buffers at a given level of processing. In addition to the limited number of buffers, we assume the control region can manage only a small number of modules. For example, 50 semantic modules might exist, each specializing in a given class of words, e.g., for categories such as animals or vehicles. Nevertheless, if the controller can remember only the four most active buffers, the number of active semantic buffers would be effectively only four buffers, regardless of the total number of modules.

Based on our interpretations of empirical literature, the number of active buffers seems to be in the range of three to four elements. In his reviews of how well Miller's (1956) magic number had withstood the succeeding decade and a half, Broadbent (1975) marshaled strong converging evidence to suggest that the magic number is closer to three than to seven items. Here we highlight only a select sample of such evidence. First, when people are asked to group common objects together, the modal category size is two or three. Second, when people are asked to divide strings of letters or digits into smaller groups, rehearsing in groups of three speeds learning better than other group sizes (see McLean & Gregg, 1967; Wickelgren, 1964, 1967). Third, pauses during free recall suggest that items are chunked into groups of no more than three or four items. Finally, mathematical analysis of hierarchical organizations suggests that chunks of three or four items provide the most efficient hierarchy for variable-order searches (Dirlam, 1972).

The present view is compatible with Baddeley and Hitch's (1974) concept of multiple buffers or slave processors. If material is presented so that some of the information can be stored in the articulatory loop, visual-spatial store, and motor system, then the working memory system will exhibit a larger memory and a lack of competition among memories. Two recent sets of experiments illustrate the benefit of utilizing multiple buffers. Frick (1984) tested subjects' digit spans auditorily, visually, and using a combined auditory and visual presentation. When the first four digits were presented visually and the remaining digits auditorily, subjects' digit spans exceeded an auditory baseline measurement by three digits. In a second set of experiments, Reisberg, Rappaport, and O'Shaughnessy (1984) demonstrated how motor memory can be used to increase the overall holding capacity of the working memory system. Subjects were taught a simple coding scheme that enabled them to store numbers in a finger-based motor program. In the

first of six experiments, subjects were able to use this coding scheme to increase their digit spans by 33% over baseline. In subsequent studies these authors extended these findings and showed that by having subjects practice the coding scheme they were able to increase their digit spans by nearly 50%. We feel these studies offer strong support for the idea that multiple buffers can be exploited to enlarge the effective workspace of working memory. The present architecture requires an explicit mapping out of these buffers, and we hope that emerging physiological data concerning the structure and function of cortical regions may help guide subsequent inquiries into the nature of the regional processing systems.

A system with multiple buffers provides a much more robust processing system. If one buffer is disturbed, information can be reloaded from the previous buffer. For example, in Fig. 4 the stimulus code for "CAT" is buffered at four levels. Even if three of the four levels are cleared, the system can still reload the output buffers to output the information.

C. Coding Item and Order Information

In the present architecture, item and order information are coded in different ways, resulting in differential sensitivity of the two types of information. In STM experiments, order information is often lost more rapidly than item information, and there is some systematicity to what is lost when (see Estes, 1972). In his pioneering work on STM, Conrad (1959, 1960, 1964) was among the first to systematically document errors of omission, transposition, substitution, and serial-order intrusion. Healy (1974, 1982) devised methods to experimentally separate the retention of item and order information. Healy (1974) showed that the relationship between order errors and serial position differs from that between item errors and serial position. Order errors (transpositions) reflect both primacy and recency effects; item errors (intrusions and omissions) reflect mostly a primacy effect. In addition, Healy showed that when a temporal sequence was to be recalled, subjects made a large number of phonemic confusion errors. In contrast, when a spatial sequence was to be recalled, subjects made no more phonemic confusion errors than expected by chance. Healy (1982) extended this methodology and provided additional evidence in support of independence of item and order information.

Order information may be coded by the buffer position, memory in the controller, or as a small part of the code if the module is storing a context-sensitive code. Item information is stored in the vector of the module. The learning mechanisms implemented in the present model can only associate vector code information. There is no LTM of control-type information. For example, a context vector may, through the use of fast weights, associate the vectors in a set of modules to the context but not to the control-type

information encoding the arrival order. This would enable context-based vectors to be reloaded (see below). However, the sequential-order information coded in the control regions may not be restored. This results in the expectation that order information may be lost rapidly. The loss of order information is not a serious problem if the next level of processing has a code for the full set of items including order. To illustrate, consider that at the word level "CAT" is a single code which could be associated to the current context, remembered, categorized, and maintained via the autoassociative processing within the module (see Schneider & Mumme, 1987). Now consider a novel letter string, "CET"; it does not have a stable code, and its order information is more dependent on the state of the control memory of the letter level which is not reset by context.

D. Order of Output

The retrieval of information from buffers is likely to occur only in a forward sequential order. In order for the central control structure to request the next item, it would need to send a NEXT signal. This could be communicated with a single binary signal, requiring no memory in the central control structure. We assume the local controller can maintain information concerning the last item transmitted and respond to the NEXT request by transmitting the next most active element. In order to support random-access retrieval of one of N buffer elements, there must be at least $\log_2 N + 1$ request signals (the extra 1 indicates a random-address request). In addition, the central control structure would have to maintain information about which modules were active and which had already been read. These sorts of requirements begin to make the central control structure a homunculus, with little processing benefit, given how rarely random access is needed. A more likely architecture would include TRANSMIT, NEXT, and RESET control signals (see Fig. 3B). This would require little memory in the central control structure and allow retransmission of messages or sequences of messages that were not well received, e.g., as in the case after a message is transmitted and no other module in the network recognizes the message.

The forward sequential coding of information at a level of processing can account for the difficulty humans have with digit canceling and reverse digit-span tasks. In a standard digit-canceling task, a series of digits is presented one at a time, typically at a rate of one digit every 0.5, 1, or 1.5 sec. The subject's task is to cancel a digit either when it first appears (immediate digit canceling), or some number back in the series. In a one-back task, e.g., the subject might see a sequence such as 7, 3, 8, 2, 4, and 5. The task would involve pressing (canceling) the number 7 when the 3 appeared, 3 when the 8 appeared, etc. Similarly, in two-, three-, and four-back conditions, the subject responds by canceling the appropriate digit back in the series while updating his or her memory.

Within the current architecture, the one-back task is easy. Whenever an input occurs, the digit is stored in the speech output buffer. When the next digit occurs, the previous digit is output from the speech buffer while the current digit is entered into the auditory buffer. The two-back task becomes much more difficult. When the third digit is input, the buffer pointer must be reset to the first position and the first digit output, then that position must be cleared (without clearing the second position in the buffer), and the third element must be added to the buffer. This would be a very difficult task for a control structure that evolved to handle forward sequential coding. Errors of loosing track of position would be very common. Humans frequently loose track of item and order information in digit-canceling tasks. Generally, the farther back in the series they must cancel, the more frequently they loose track of such information. For example, in an early variation of a canceling task using lights rather than digits, Kay (1953, in Welford, 1968) found that at the rate of one new light every 1.5 seconds, subjects' average correct performance declined from 95% (one back), to 67% (two back), to 47% (three back), to 35% (four back). Furthermore, when Mackworth (1959) used this same procedure and established presentation rates on the basis of ability to achieve an 80% accuracy criterion, she found subjects needed progressively more time to perform as they canceled farther back in a series, e.g., at one back they required 1 sec between items, at two back 1.6 sec, at three back 2.4 sec, and at four back 3.8 sec. We encourage the reader to get a sense of the difficulty of this task by having a colleague read a series of 10 random digits at a rate of one/sec and then try to respond to the digits two back in the series.

The use of forward sequential coding would also explain the difficulty of outputting sequential information in reverse order, as in reverse digit span. The last item could be output by retransmission of the last input. However, to get to the second-to-last item, the list would have to be output from the beginning until the second-to-last item was reached. Normal forward digit span is typically about seven items (Lyon, 1977), whereas reverse digit span is typically two items less (Starr, 1929; Anders & Lillyquist, 1971). The current control structure could do reverse digit span by outputting the last item, then outputting from the start of the list until the next item to be output matched the last item output before the reset. With a buffer size of four, plus another single storage buffer, a reverse digit span of five would be possible. However, the reaction time to output the items in reverse order would be slow, particularly for the items near the end of the input list but not the last item (see Anders & Lillyquist, 1971).

E. Rehearsal Loops

In the present architecture, a rehearsal loop can be implemented if a message can be passed between a series of modules and the message

transmission results in invertible codes. For example, in Fig. 3 a rehearsal loop could be established between the auditory input buffer and the speech output buffer. A vector code for a word could be transmitted on the innerloop to the speech buffer. Then the speech buffer could transmit the word to the auditory input buffer. In order for rehearsal to be successful, the code must be invertible. An invertible code is one in which a one-to-one mapping exists between each code set. This is the case for speech; for every auditory representation of a spoken word, there is a motor representation of the spoken word. Hence, given that each code is associated uniquely to the other, it is possible to map to the other. Invertible codes are probably not the norm for communications on the innerloop. Most communications probably involve many-to-one mappings, e.g., there are many color patterns that map to the word *red*. The speech buffer might be able to bias the visual system, but it cannot produce a clear code. One would expect to be able to develop rehearsal loops where the codes are invertible. For example, training might allow one to build an auditory-to-motor rehearsal loop by mapping each word to a specific motor movement and each motor movement to a unique auditory code.

The present architecture illustrates many of the buffer memory phenomena associated with working memory. There are buffers in many levels and in many regions of processing. The primary purpose of these buffers is not to provide a STM store, but rather to enable the robust processing of sequential input and output of temporal or spatial information. Different regions may have different numbers of buffers, levels, and time constants.

V. Context Effects, Proactive Interference, and Release from Proactive Interference

A system having only a short-term buffer memory and a permanent LTM would probably not survive in a world with interruptions. The buffer memory would work well as long as the incoming stimuli were continuously being processed and the buffers were enough to contain and operate on relevant information. If the buffers were ever flushed, however, the system would loose its orientation in time and space and would have to search the environment to determine where it was and what it was doing. Having a system with only buffer memories would be like operating a computer system with only active memory and no backup tapes or disks. If the power were ever lost, the system would have to begin from scratch. LTM would help, but it is important not to use LTM as a working memory. This is because the faster LTM is changed, the greater the likelihood that retroactive interference will distort all the previously stored LTM, making it useless (see below and Fig. 6).

A. Episodic Memory

A system that includes a context or episodic memory is a much more stable system. As an analogy, consider how computer centers use periodic backup procedures. Every night the system backs up all of the files stored in the system. If there is a failure, such as a system disk crash, the most that is lost is one day's work. The context module in the present system provides an analogous sort of backup. Every time a new vector is stored in the innerloop of modules, the connections between the context vector and the message vectors change so that the context vector can reevoke the message vectors. The context vectors may be changing continuously and providing a time stamp of the current contents of the system. Therefore, it might be possible to reestablish past contexts that occurred at various previous states, e.g., 30 seconds, 5 minutes, 1 hour, 1 day earlier.

A context or episodic memory provides three critical benefits to the system. First, it makes the system much more robust. If the active memory buffers are flushed to process some critical event, the buffers can be reloaded as a result of the context vector reestablishing the previous contents of the buffers. Second, it provides the features of temporal orientation that Tulving (1972, 1983, 1984) cites as benefits of episodic memory. These include maintaining space–time orientation, allowing time-tagged judgment and retrieval, and providing autobiographical memory. Third, the episodic memory may allow the use of remindings, i.e., remembering a previous sequence of actions, (see Ross, 1984; Schank, 1982) to enable the performance of procedures before procedural knowledge has developed.

Additional evidence for the potential value of context comes from the study of memory deficits. Some amnesiacs, such as HM, can perform STM tasks and learn procedural tasks, but they cannot recall or recognize words presented a few minutes before (Cohen & Squire, 1980; Cohen, Eichenbaum, Decacedo, & Corkin, 1985). Perhaps HM's pathology illustrates how debilitating processing might be if one had only a short- and long-term memory system without any type of time-tagging to maintain temporal orientation. If HM is distracted from a simple task, the task must be reexplained to him again from the beginning. HM cannot survive in everyday life and must be under close supervision at all times. Because he cannot remember where he is or why he is there, he is prone to wandering and getting lost.

B. Proactive and Retroactive Interference

In a context memory system, storing knowledge in fast connection weights will show rapid knowledge acquisition, but it will also exhibit severe proactive and retroactive interference. By using the fast learning rate, knowledge is quickly acquired but quickly forgotten. Figure 6 illustrates this

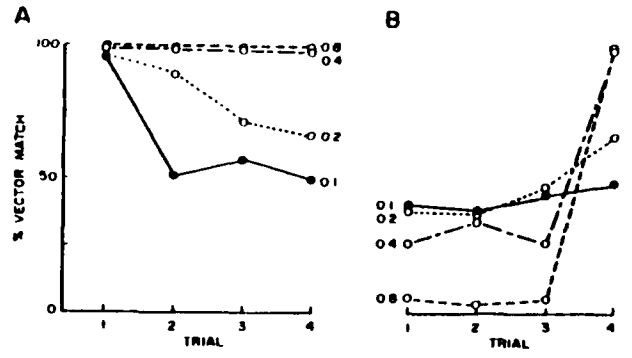

Fig. 6. Retention and recall; the effects of proactive interference, left, and retroactive interference, right. The numbers refer to the learning rate for the delta-learning rule used during acquisition. The vector match illustrates how accurately the retrieved vector matches the to-be-learned vector (0% is chance, 100% a perfect match). In contrast, the learning curve shows retention for that pattern on that trial before the next trial is begun. The retention curves represent recall of all four patterns after four learning trials. The context vectors were correlated .9 between trials. The input pattern was activated for a burst of five iterations. Autoassociative feedback was on for all iterations and the output occurred on iteration 8.

relationship for a simple simulation. The network was trained to associate four random output vectors to a changing context vector. The context vector was a 200-element binary vector. From one trial to the next, 10% of the elements were resampled, resulting in the context vectors having a correlation of .9. The learning curves (Fig. 6A) show how accurately the context vectors reproduced the desired response vectors. Figure 6A shows the information available at the end of each trial. This is similar to what would be expected in a STM experiment with a long period of distracting material, e.g., as in the Peterson and Peterson (1959) experiment with 18 sec of counting backwards by 3s. Figure 6B shows end-of-list recall for all four associates. The system presented the four context vectors for the four previous trials and compared the output to the originally learned vectors to determine the percentage of vector match. This would be similar to a free recall experiment for a four-item list with a distracting task to eliminate any retrieval from buffer memories.

The first association is learned nearly perfectly over a wide range of learning patterns. This is because in a new connection matrix there are no previously learned associations that activate incorrect connection patterns. In the present architecture, a pattern activation involves evoking a pattern

and then categorizing the pattern (via autoassociative feedback) to be one of the previously learned patterns. A vector only one one-twentieth as strong as another association, e.g. learning constants of .05 and 1.0, produces nearly identical recall on Trial 1, e.g., 92.5% match after a learning trial with a .05 learning rate, 95% for .1, 98.5% for .4, and 100% for .8. Performance resembling a first trial occurs when either the previous connection weights are all zero or when the current retrieval vector is orthogonal to the previously learned vector (see below and Fig. 8). For example, in the current simulation, waiting 20 trial periods reduced the correlation to .12 and produced learning performance very similar to Trial 1 performance.

Small learning rates (i.e., slowly changing weights) show serious proactive interference effects (see Fig. 6A). This is because the previously learned patterns interfere with the current pattern. If these patterns point the vector toward previously learned patterns, the feedback will retrieve combinations of old patterns as opposed to the current pattern. With a large learning rate, the current learning trial will swamp the effects of the previous learning trials, e.g., the .4 learning-rate condition in Fig. 6A. If the purpose of the memory is to reload the contents of the last trial, a high learning rate is beneficial.

Large learning rates (i.e., fast changing weights) show retroactive interference effects. Figure 6B shows the retention after learning four patterns. For a large learning constant (e.g., .8), Trial 1 retention is nearly at chance after only three intervening trials.[11] Note that the ordering of retention conditions on Trials 1 and 4 are opposite. The highest learning constant (i.e., .8) produced the worst Trial 1 and best Trial 4 retention (see Fig. 6B). Fast learning develops the association in a single trial but at the expense of forgetting everything learned previously. Trial 4 was the last trial, and hence retroactive interference was not a problem. In sharp contrast to the condition with a large learning constant, the condition with the smallest learning rate (i.e., .1) showed the worst Trial 4 retention and the best Trial 1 retention of any of the learning rates. All conditions show a recency effect, with the effect being larger and involving fewer trials with the larger learning rates. The smallest learning-rate condition shows some evidence of a primacy effect. The retention data show that if the purpose of the memory is to retrieve information learned many trials previously, smaller learning rates are preferred.[12]

The differences in proactive and retroactive interference for small and large learning rates illustrate the benefit of evolving a system with multiple learning rates. The large learning rates provide for quick acquisition and

[11] There is a floor effect, in that very low learning rates do not modify the association matrix and illustrate poorer learning and retention. For example, in the current simulation a learning constant of .1 showed better learning and retention than that of a learning constant of .05.

allow the system to perform the task, while the small rates encode information for later retrieval. If the learner practices the task extensively, the small learning-rate connections (slow weights) will acquire the information before the large learning-rate connection associations deteriorate due to retroactive interference.

The literature on proactive interference effects in STM research is consistent with the existence of a context memory with fast weights. Recall that the first trial of a STM experiment is nearly perfect (Keppel & Underwood, 1962). Baddeley and Scott (1971) found effectively no decay for the first STM trial for delay intervals ranging from 5 to 36 sec. Subjects' performance declines markedly from the first to the fourth trials in a STM experiment, e.g., reducing from 68% to 25% (Goggin & Wickens, 1971). This proactive interference is a temporary phenomenon. In the present architecture, the longer the delay between trials, the more the context connection weights decay and the context vector changes, resulting in a greater release from the effects of proactive interference.

C. Release from Proactive Interference

Research on release from proactive interference (see Wickens, 1970) may illustrate the local nature of proactive interference (PI) effects. If subjects are required to remember sets of three words from a single taxonomic category in a STM task, accuracy drops dramatically between the first and fourth items of the list. However, if the next word is selected from a new category, performance increases substantially—almost to the level seen on the first trial. This improvement is called the release from PI. Figure 7A illustrates data from Loess (1968) showing this effect. Subjects were presented sets of six words from one of four taxonomic categories. Group 4A received items alternately from the four categories, whereas group 4S received six items sequentially. Notice the dramatic peaks in the solid lines when the category was changed in the 4S condition, i.e., on Trials 7, 14, and 19. These peaks illustrate how switching categories can produce a release from PI. Figure 7A also shows a strong PI interspersed between the repetitions, as shown by the drop-off in the 4A condition on Trial 5.

The connectionist/control architecture will produce a category release from PI if different semantic categories are represented in different modules within the network. The buildup of PI is a result of storing multiple patterns in one set of connection weights. To illustrate, if one module codes vehicle information and another codes animal information, then there are two sets of connections (or association matrices) between the context and the modules containing the semantic information. In our current simulations, storage results when a transmission is succeeded by a follow-on transmission from the receiving module (see above and Schneider & Mumme, 1987). Hence, learning only occurs at the intersection of those

The physiological evidence on rapid and slow learning (Mishkin, et al., 1984) suggests that primates have evolved a two-speed learning system.

effect as a result of repeating the same category, even if three other sets of category items are

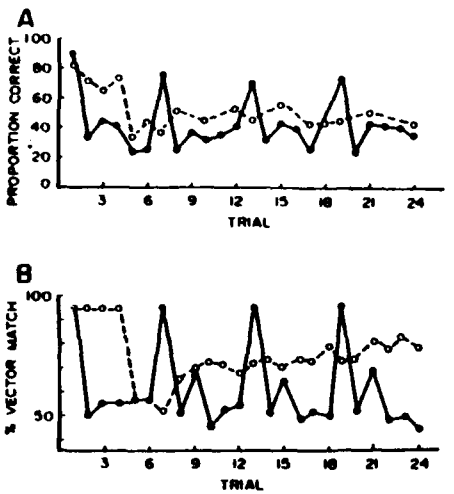

Fig. 7. Release from PI. A. Human data from Loess (1968). B. Results from the simulation. The solid lines are the 4S conditions (massing all the items for a category). The dashed lines are the 4A conditions (alternating all four categories before repeating an item).

fibers which input a message just before the module outputs a message. In a release from PI experiment, all of the modules could potentially receive a context message, yet only the module containing the rehearsed item would output a message. This means that only connections within that module would be changed.

In the simulation of the category learning experiment, the model was presented 24 items from four categories. The four categories were represented by four sets of different association matrices. A word rehearsal was assumed to involve a transmission of the word from the auditory module to the semantic module, and a transmission of the semantic code to the auditory module. Prior to every transmission the context vector was transmitted. To simulate time delay, 10% of the context vector was changed randomly on every trial. The learning constant was .1. On each trial the word and the semantic vector were associated to the context. The context was then used to retrieve the vectors.

Figure 7B plots the percentage of match between the retrieved and the to-be-learned vector. Both the word and the semantic vectors were retrieved. The percentage of vector match plotted in Fig. 7B represents the maximum

of the word and semantic vector. This produces slightly higher recall for the first few items than when only the semantic vectors are used. The probability of recall is a monotonic function, e.g., logistic function, of the percentage of vector match. The actual probabilities depend on vector size, number of vector codes, feedback, and noise level. With appropriate parameters, a 50% match could produce a 20% recall, making the simulation data comparable to the Loess data.

The simulation produces five qualitative features of release from PI as seen in the Loess (1968) data. First, there is marked PI for repetitions of words in the same category. This occurs both for the 4S condition (e.g., the difference between trials 1,2,3, . . . 6) and the 4A condition (e.g., the difference between Trials 1-4,5-8, . . . 21-24). The proactive interference is a result of interference from the previous learning trials (see discussion of Fig. 6). Second, there is a sharp increase in accuracy or release from PI when the category is changed in the 4S condition, i.e., on trials 7, 13, 19, 23. This is because a new category is assumed to be stored in a different semantic category with a different set of connections to context. Third, excluding the release from PI trials, the 4A condition showed better recall than the 4S condition (Trials 2-6, 8-12, 14-18, 20-24). This occurs in the simulation because the context vectors are correlated .65 in the 4A condition and .9 in the 4S condition. Remember that in the 4A condition a category is repeated every fourth trial, leading to an average correlation of $.9^4$ and $.9^1$ in the 4S condition. Fourth, the second exemplar from each category in the 4S condition (Trials 2, 8, 14, 20) shows particularly poor performance relative to the first and third exemplars of the category. This occurs as a result of the very good learning of the first exemplar causing more proactive interference on the second trial. The poorer second-trial learning causes less proactive interference on the third exemplar, thus producing better performance on Trials 3, 9, 15, and 21 relative to the predecessors.[11] Fifth, the second repetition of the categories (Trials 5-8 in the 4A condition) is inferior to the preceeding or succeeding set of categories. This is again due to overshoot from learning the first item of the category.

The release from PI results provide an indication of how large the effective working memory might be. Each association matrix between the context module and every other module could store one or more vectors. If the context module were to produce orthogonal codes (see Kohonen, 1984), a matrix could store as many vectors as there are fibers. To illustrate, if there were 100 modules in the innerloop and 10 fibers from the context module to the other modules, the theoretical capacity could be as high as 1000 codes. To the extent that the codes are not orthogonal, capacity would be reduced

[11] This learning overshoot effect on the second exemplar does not occur for larger learning rates (see Fig. 6A Trial 5 versus Trial 3).

accordingly. Human data suggest that probably only one vector can be tied to the context vector for each module in the short term; and over extended periods (minutes), it may be possible to store several sets of vectors. Data from three sets of experiments are compatible with this view. Peterson and Gentile (1965) showed no effects of PI for the first items of a block when the blocks were separated by 91 sec; Loess and Waugh (1967) showed no effects of PI beyond 120 sec; and Kincaid and Wickens (1970) showed the greatest reduction of PI after 45 sec and a reduction of about 74% after 120 sec. In sum, these data suggest that a combination of a changing context vector and perhaps decaying weights allows the system to store a new set of codes every few minutes.

The present connectionist model has some similarities to the context-retrieval procedures present in the search for associative memory model, SAM (Raaijmakers & Shiffrin, 1980,1981). In the SAM model, retrieval is based heavily on having items associated with a list context and interitem associations. The model predicts a variety of LTM phenomena, including serial-position effects, list-length and presentation-time effects, temporal aspects of free recall, part-list cuing, and cued recall of paired associates. The current connectionist model provides a mechanism by which a cuing model such as SAM might be implemented in neural-like hardware.

D. Recency and Primacy Effects

The use of context storage also provides an interpretation of recency and primacy effects in LTM. Tzeng (1973) had subjects perform a free-recall task in which subjects learned four 10-word lists. Each word was presented for 1.5 sec followed by 20 sec of counting backwards by 3s. Tzeng's data showed a clear recency effect at end-of-list recall and end-of-session recall. The existence of a recency effect following 20 sec of interfering activity violates expectations of basic buffer models.

This result would be expected, however, if the context at the time of recall were used as a retrieval cue. Retroactive interference would produce a positive recency effect as is illustrated in Fig. 6B. Since there is typically a delay between the end of one list and the beginning of the next, there will be less information stored with the context vectors active at the ends of the lists, resulting in both primacy and recency effects.

Some authors tend to interpret such long-term recency effects as data against the existence of a STM buffer (e.g., Tzeng, 1973; Crowder, 1982), primarily on the basis of parsimony. With the connectionist/control architecture, we expect both short- and long-term recency effects to exist and to have quite different mechanisms. These two mechanisms make different predictions that can be tested. First, increasing the duration of the interfering task should increase the recency effect for long-term retrieval and decrease it for short-term retrieval. Second, combining a short interfering

task at a normal presentation rate in a free-recall task, e.g., performing four digits of a two-back digit-canceling task, should greatly attenuate short-term recency effects.

E. Overloading STM

Context storage enables the network to perform reasonably well, even in situation in which the short-term or buffer memory is heavily loaded. Klapp, *et al.* (1983) present compelling evidence that such loading can occur without catastrophic effects. When subjects were required to retain letters in a span task, this activity did not interfere with their abilities to judge the correctness of greater-than/less-than statements (Experiment #7). Similarly, Klapp and Philipoff (1983) found that subjects could retain letters and concurrently process digits in a missing digits task. Logan (1979) has similarly loaded STM with six digits and found little interaction with number of letters searched in a visual search task.

Such results are quite incompatible with the view that working memory has only seven slots. However, in the connectionist/control model, a context storage mechanism can account for these effects. The subject first rehearses the STM list. This connects the context vector to the rehearsed codes. The subject then performs the embedded task, perhaps processing information in the same buffers, but not rehearsing information in the buffers. After the embedded task is completed, the subject activates the context vector that was present at the time of rehearsal and activates the rehearsed items for sequential output. If subjects are required to rehearse similar material in the same modules, interference should occur due to retroactive interference from the second set of material. Mutual interference does occur, despite early rehearsal, if the embedded task requires item and order information to be retained (Klapp *et al.*, 1983, exp. no. 8; Klapp & Philopoff, 1983). Recall (see above) that context coding may provide only weak coding of order information.

In summary, the present architecture can accommodate many of the effects of context and proactive interference. Moreover, we submit that some type of context-based storage is needed to guarantee robust information processing, since a system with only active buffers and a slowly changing LTM is inherently labile and unlikely to survive in the real world. The proposed context-storage system provides interpretations for (1) how episodic memory might function; (2) how the effective working memory might be far larger than four to seven elements; (3) why there is a high level of retrieval and lack of interference effects on the first trial of a STM task; (4) why release from PI is expected if different semantic categories are represented in different modules; (5) how LTM recency effects might arise; and (6) how an information-processing system might still perform even when STM is heavily loaded.

Nor did it impair the performance on a modified Sternberg-type scanning task.

VI. Skilled Memory, Mnemonics, and Levels of Processing

If working memory includes a large number of regions, levels, and buffers, plus context storage and attentional control, then there are likely to be good and bad strategies for using it. The skilled use of working memory involves allocating and substituting resources to maintain information. We assume the overall resource pool is quite differentiated, with different resources varying in terms of what type of material can be stored, the time required to store the material, PI effects, retrieval time, trace decay, and the robustness of the storage. In terms of the model, we propose that storage is dependent on which modules are active, what the input vectors are to the modules, what codes are in the modules, and whether a module transmits messages after an input.

A real-world example of the use of skilled memory comes from the study of a waiter, dubbed JC, by Ericsson and Polson (1987). JC was reported to be able to remember over 20 complete orders without using an external memory aid. In controlled experiments, Ericsson and Polson found that JC was indeed able to perform simulated order tasks with high accuracy. They speculated that JC used retrieval structures analogous to those used by experts in digit span (see below). To remember a sequence of orders, JC rehearsed the first four orders and developed a well-integrated structure for them before trying to remember the next four in the sequence. Ericsson and Polson characterize this structure in terms of a matrix with one dimension representing the relations among items comprising an order. JC associated the customer's entree to context features (a customer's face) by *constructing* an interactive representation. The other dimension represented the items into categories, i.e., into entrees, meat temperatures, kind of salad dressing, and kind of side dish (starch). To create a unique retrieval cue for salad dressings, JC encoded them by their first letter; e.g., to remember four different salad dressings, JC encoded blue cheese, oil and vinegar, and thousand island as *B-O-O-T*. To remember the items in the other three categories, JC relied on different encoding schemes. Temperatures were encoded as spatial patterns, starches as serial patterns, and entrees in terms of repetitions and patterns that resulted from partitioning orders according to cost. JC developed within-category relationships dynamically, i.e., as he was given a new order he used the different category labels to know where to put a new item and then proceeded to order the old and new items into a coherent structure. Finally, it should be noted that when JC recalled dinner orders he always did so categorically. In the following section we offer a rudimentary framework of rules for thinking about how to develop skilled memory such as JC's within the proposed architecture.

A. Rules for Skilled Memory

The connectionist/control architecture suggests five rules for the skilled use of working memory. These rules describe methods of capitalizing on the relative strengths of different types of memory to maximize storage.

RULE 1: *Use multiple buffers to increase the skilled use and capacity of STM.*

If a subject is required to perform two tasks, X and Y, and task X can be performed in buffer A, and task Y can be performed in buffers A or B, then task X should be put in A and task Y in B. Many of the experiments on STM load suggest this type of allocation scheme. To be able to use buffering strategies effectively, one must first learn how to alter them depending on situational demands. Different task mixtures will be performed better with some allocation policies than others. For example, digits in a spatial relationship might be stored spatially, e.g., as a visual image of a grid, or verbally, e.g., as the proposition 5 to the left of 8. If the subject must perform a concurrent tracking task, it would be better to store the digits verbally. But when the concurrent task requires auditory processing, it would probably be better to store the digits spatially (Baddeley, Grant, Wight & Thomson, 1974).

If a buffer is likely to be disrupted by irrevelant input, the information should be shifted to a buffer isolated from that input. The unattended speech effect (Salame & Baddeley, 1982) and the suffix effect (Crowder & Morton, 1969) suggest that irrelevant input can disrupt auditory input buffers. To achieve the unattended speech effect, Salame and Baddeley had subjects perform a visual digit-span task with an irrelevant auditory word presented with each visual digit. The irrelevant words reduced digit recall by 46%. In contrast, bursts of white noise produced a much smaller decrement of 19%. To achieve the suffix effect, a subject reads an irrelevant verbal item at the end of a string of digits or words in a span task. The irrelevant verbal item invariably reduces recall of the last few items of the list. In one such experiment, Ayres, Jonides, Reitman, Egan, and Howard (1979) showed that accuracy of the last item dropped from 88% to 32% due to the addition of a word suffix. These effects illustrate that with a continuous verbal input stream, it would be beneficial to recode information spatially and to maintain it in a spatial buffer insulated from the input stream.

There are also alternatives to storing information in the form of active codes in buffers; codes may be associated to context vectors or other information vectors. These two types of associate may store information at different rates, show differential effects of proactive and retroactive interference, and may decay at different speeds. The speed with which the context vector can be changed is probably slow relative to the speed with which other vectors can be changed.

Context storage provides a method for rapidly associating information to the current context. It has the potential of being an automatic mode (in the sense of Hasher & Zacks, 1979) of storing information. If the context vector is transmitted periodically, the connection weights can change such that transmitting the context vector can reload the vectors that were present in the network at the time of the last context transmission. If the context vector involves fast-changing weights, learning will be quick but proactive and retroactive interference will limit the usefulness of the storage.

RULE 2: *Store codes in unique modules that will not be reused until the context changes.*

This tactic typically involves coding information elaboratively. Storage in the network occurs after a module receives and transmits a vector. To store information in the connection weights of the low-reuse module, the code from the low-reuse module must be transmitted. The benefit of elaborative rehearsal illustrates this type of storage. A subject could learn a word list by verbalizing the words of the list repeatedly. In this case the context weights would be altered for every word. The buildup of proactive and retroactive interference would eliminate any benefit from context-based recall after the first few trials. In contrast, if the subject were to code each word semantically, different modules would code different words. Remember, storage occurs after the transmission of a message. To associate the context vector to the semantic module, the network must transmit the semantic code. To semantically code the word *cat*, the subject might activate and transmit the concepts "a warm furry object that purrs." Context would now be associated with that code in the semantic module coding animal-like features. If no other word were stored in that module with the same context vector, there would be no problems with proactive and retroactive interference. Therefore, to use context memory skillfully, one should try to code each word . . a unique module. If a second word were to evoke a code in a module that had already stored a code, then that vector should not be transmitted, and perhaps the second-most-activated semantic module should be transmitted.

From the present perspective, elaborative encoding and release from PI illustrate the same effect. Simple verbal repetition of items is like Peterson's (1968) 4S condition (see Fig. 7A, Trials 2–6), in which the same module is reused for all the words. Elaborative encoding is like the 4A condition (see Fig. 7A, Trials 2–4), in which different modules are used on different trials. The differences between the 4A and 4S conditions (69% versus 34%) are comparable to the differences between elaborative and rote verbal rehearsal.

Training may be necessary to establish strategies of the central control system to identify unique modules to transmit, and hence, to store context information. To later retrieve context-stored information, the context vector would have to be transmitted, activating codes in a series. The context-activated

semantic vectors could then be transmitted to the speech region for verbalization of the words. Note that this system codes order information poorly. There is no inherent coding of order; the system simply has a list of codes associated to a context vector. However, if the context code changes in some continuous manner over time, the strength of connection to different contexts may provide a coarse time stamping.

RULE 3: *Develop retrieval cues that are clear, distinct, active, and related to the material to be retrieved.*

The problem of PI results from associating several output vectors to a single, or several highly related, input vectors. We have assumed that the context vector is a slowly changing vector requiring perhaps two minutes to change substantially; recall that most PI effects in STM procedures dissipate in less than two minutes (see Peterson & Gentile, 1965; Kincaid & Wickens, 1970). By switching attention among a list of well-known items, the subject could rapidly alter what vectors are active in the network. If these vectors were dissimilar, i.e., orthogonal, there would not be a buildup of proactive or retroactive interference.

Mnemonic techniques generally provide a list of well-known items to associate information to (see Bower, 1970; Bellezza, 1981, 1982). For example, in the peg-word system, the subject activates a series of images of concrete objects in a list, e.g., one-bun, two-shoe, three-tree, while the method of loci involves committing well-known places to memory. The subject then associates each new word or phrase to one of the images in the list. At recall, the subject sequences through the peg words or locations and retrieves the words associated with each retrieval cue.

Using mnemonic strategies would result in better memory in the connectionist/control architecture. To associate a word to a peg requires transmitting the peg-word code, and then transmitting the to-be-remembered code. If the subject repeats only the to-be-remembered words, then the only retrieval code would be the context and perhaps the previous word on the list. Since the context is a slowly changing code, multiple associations build up PI. This interference makes retrieval unlikely if more than a few words are associated to one context, e.g., learning more than three words every two minutes. The advantage of using a mnemonic is that learners can alter the code rapidly by changing the object they are attending to. To the extent that these cues provide orthogonal codes, PI should be greatly reduced. If learners use a well-learned sequence, as in the peg-word or loci mnemonics, they could retrieve the ordered set of retrieval cues. Then, the prelearned retrieval cues and the context could be used to retrieve the newly learned codes.

Figure 8 illustrates the importance of using dissimilar retrieval cues in recall. In the simulation we associated a list of four input vectors to four output vectors. Then the model recalled the output vectors using the input

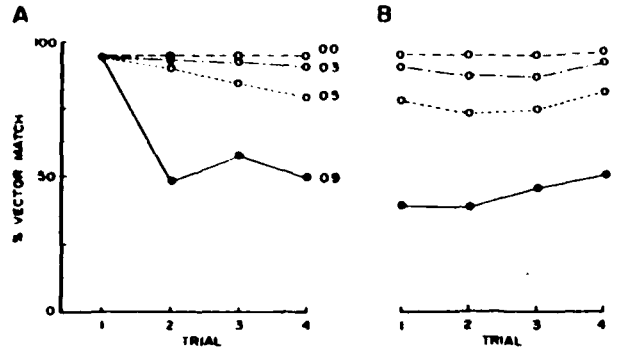

Fig 8. Similarity of retrieval cues on learning and retention. The numbers represent the sequential correlation between vectors during learning. The learning constant was.1 in all conditions. (See caption, Fig. 6).

vectors as retrieval cues. Note the marked recency effect when the vectors are correlated. If context is a slowly changing vector, it would have a high correlation from one word to the next, and it would show interference effects similar to the curve with a correlation of .9 between vectors. In contrast, if the peg words were uncorrelated vectors, they would provide recall similar to the curve with a correlation of 0.

Reducing the similarity of retrieval cues both reduces the buildup of *PI* (Fig. 8A) and increases retention (Fig. 8B). This suggests that the effectiveness of spacing in producing a release from PI (Wickens, 1970) and mnemonics have a common mechanism of increasing performance by providing more dissimilar retrieval cues.

The use of mnemonics for both intermediate and long-term retrieval suggests that the association of information messages involves both fast and slow weights. The ability to quickly associate new material to a loci retrieval structure or to a peg-word system and to have those associations decay over a period of hours suggests the presence of fast weights between the information vectors. Using the method of loci to remember long stories months later suggests the involvement of slow weights. The SAM model (Raaijmakers & Shiffrin, 1980, 1981) illustrates how rapidly modified associations in LTM might be used. *In this model, every time a word is attended, the strength of* association of the word context and other words active in STM is increased.

Retrieval cues that are related to the to-be-retrieved information allow easier recall of information than unrelated cues. For example, if a category name were used as a retrieval cue, the preexisting associations between the

category and the exemplar would greatly reduce the amount of learning that needed to occur. The category name would evoke most of the semantic features of the word and in so doing identify which module contained the associated information. The context input need only bias the module to resolve which member of the category to retrieve. The fact that words from a given category are clustered in free recall (Bousfield, 1953; Bousfield & Cohen, 1955; 1956) suggests that multiple words benefit from the same retrieval cue or that they reside in the same module. Humans can quickly associate a few examplars to a number of categories with little evidence of interference (Mandler, 1967). They can also learn to retrieve lists of hierarchically organized material after short study times, e.g., learning up to 112 words after an average study time of 2 sec/word (Bower, Clark, Lesgold, & Winzenz, 1969).

RULE 4: *Use multiple retrieval cues and distribute practice.*

Most connectionist models use some variant of an error-correction learning rule (see Hinton & Sejnowski, 1986; Rumelhart, Hinton, & Williams, 1986). In the present model, we use a delta learning rule which changes the strength of association in proportion to the error between the vector evoked by the input and the desired vector (generally the vector already in the module as a result of previous processing). If there is no error, there is nothing to correct and hence no learning. Repeated associations to the same vector will typically result in an exponential reduction in amount of learning. The marginal utility of continued rehearsal of the same association decreases as a function of repetitions. However, if the subject switches to a new retrieval cue that is not associated to the output, the new cue will cause a large error so that the learning trial will produce more connection change. Associating an output to multiple input cues provides alternative retrieval paths for later recall.

Distributing practice enhances learning because of the nature of changing connection weights with an error-correction rule. Rosenberg and Sejnowski (1986) have shown that a connectionist learning model will learn a set of 1024 patterns with better retention under spaced practice (going through the entire set one at a time) than under massed practice. Massing practice is equivalent to learning with a large learning rate. As mentioned above, large learning rates are problematic because they produce greater retroactive interference (see Fig. 6). If practice is distributed, the network searches the connection space to find a set of connection weights that provide the minimum error for the total ensemble of patterns to be learned. Because the connection spaces generally involve a large number of connections, there are many possible sets of changes in the set of connections that will produce nearly the same output for a given input. By distributing practice, the error-correction rule moves the weight space to a more global minimum for the entire ensemble. In contrast, massing practice moves the

weight space toward a minimum for that one pattern (see Rosenberg & Sejnowski, 1986, for discussion).

The presence of context storage increases the importance of spacing practice and provides an interpretation for the generation effect (Cuddy & Jacoby, 1982). If the context vector involves fast-weight changes, repetitions of an item to the same context will result in a lower marginal utility for each repetition. Fast weights are valuable because they enable context-based recovery of information within the same temporal context (see above). Fast-context weights might be potentially detrimental, in that the majority of learning occurs in these weights and context may not be a good retrieval cue, either because it changes or due to problems of retroactive interference.

The generation effect illustrates how context association can harm learning. Cuddy and Jacoby (1982) used a crossword puzzle task to investigate how memory for an earlier solution would influence subsequent puzzle solving. Subjects were presented combinations of reading and construction tasks. In the reading task, the subject read each of two related words, e.g., *lawyer, court*, while in the construction task the subject read the intact word and then solved the puzzle and reported the solution, e.g., lawyer c __ __ rt. Using this procedure, Cuddy and Jacoby found that a subject's memory for an earlier presentation of an item can influence subsequent problem solving at least a few minutes later. In addition, they found when a problem was repeated so that its repetition resulted in greater processing, memory for an earlier presentation was less accessible. In the present model, presenting the word earlier would build an association between the context and the puzzle word. The prior presentation of the word would reduce the amount of attention and the amount of noncontext learning the word received, even if it were attended. This type of learning effect produces overshadowing phenomena similar to the Rescorla and Wagner (1972) model.

RULE 5: *Use well-learned codes in the receiving modules.*

Within each module we postulate an autoassociative matrix that associates each learned code to itself. As mentioned above, the autoassociative mechanism is important for cleaning up noisy input and categorizing the input (see J. A. Anderson, 1983; Schneider & Mumme, 1987). The autoassociative effect provides nonlinear feedback so that similar inputs can produce dissimilar outputs (see Schneider & Mumme, 1987). This feedback also helps to maintain information in buffers (see above).

This autoassociative effect is the basis of the interaction between long-term and short-term memory. In the simulation, the effect can be removed by setting the autoassociative feedback to zero, thereby simulating the absence of within-module long-term knowledge for the trace. The recall of four paired-associates with a learning constant of .1 had an 18% vector match for a feedback of 0 and 42% match for a feedback of .4 (see Fig. 6B).

To learn arbitrary material, such as digit strings, it should be be beneficial to recode the material in a representation that is already well learned. For example, in Smith's classic experiment in which subjects recoded each sequence of three binary digits into one octal digit, immediate memory span increased from about 12 to 40 digits (see Miller, 1956). Similarly, Slak (1970) has shown that by acquiring a recoding scheme to translate strings of digits into groups of pronounceable CVCs, one can improve performance markedly on a wide range of digit-based tasks, including serial learning, free recall, recognition, and span tasks.

The research on practice effects in the development of skilled memory (Chase & Ericsson, 1981, 1982; Ericsson & Chase, 1981) illustrates the use of all five rules of skilled memory. Chase and Ericsson had their subject SF perform a digit-span task for 230 hours. SF was presented digits at a rate of one/sec and then asked to recall the digits in serial order. Digit span was defined as the number of digits the person could repeat back correctly 50% of the time. Over the 230 hours of practice, SF's digit span increased from 7 to 79 digits. Chase and Ericsson argued that this skill was accomplished as a result of (1) associating new material to the material in LTM, (2) storing information in a "retrieval structure," and (3) increasing the speed of encoding and retrieving items with practice. SF's strategy was to buffer the input stream and to try to associate the information in groups of three or four digits.

Digit-buffering illustrates Rule 1, storing information in multiple buffers and moving the information to a lower-activity buffer, while trying to associate it to new information. SF would passively store a group of three or four digits and then encode the digit group into a well-learned code, e.g., track running times such as a world-record mile-running time set by a specific runner. This recoding illustrates Rule 5, recoding new information into stable LTM codes. SF stored and retrieved information in an elaborate retrieval structure. He would recode digits into sets of three- or four-digit groups; these groups were organized in a hierarchical retrieval structure of groups and "supergroups" of groups of digits. This retrieval structure provided both differential locations at which to store information (Rule 2: store in unique buffers) and unique retrieval cues (Rule 3: use different associations to retrieve the information). For example, storing four-digit mile-running times would not interfere with storing three-digit times for half-mile runs.

Observe that if the same buffer were not reused within a short period of time, retroactive and proactive interference would not be a problem. With extended practice, e.g., 230 hours, it may be possible to specialize additional buffers, e.g., mile-running times for the first part of the list, and thus provide more storage capacity. The retrieval structure also provides unique retrieval cues, e.g., associating in a hierarchical structure of groups and

and supergroups. After a year of practice, these cues may have become very salient and recoded internally as more orthogonal vectors."

The human working-memory system embodies subsystems that are capable of being deployed in a variety of strategies. In the current connectionist/control architecture, different strategies will exhibit a wide range of effective capacities. If the subject only uses one set of buffers, then capacity is limited from three to five codes. If, on the other hand, the subject uses multiple buffers, then capacity may be limited by the decay time of the buffers, or it may be limited to a capacity of four codes/buffer. If the subject uses context as a retrieval cue, the capacity may be limited to one code/module within the same context. If the subject attends to orthogonal retrieval cues, the capacity may be limited to one code/module for each orthogonal retrieval cue. To develop skilled use of working memory may require extensive training to utilize the best mixture of learning strategies in the face of task-specific conditions.

VII. Serial Outputs and Chunking

A. Sequential Hierarchical Output

Lashley (1951) stressed that sequential output is a very fundamental and common form of human processing. In this section we provide an interpretation for sequential output, chunking, pause boundaries, and chunk-based information retrieval. Up to this point our discussion has focused on how information reaches the innerloop of processing. Now we discuss how the higher-level codes are converted into sequences of actions. The codes feeding into the innerloop are highly compressed codes that are buffered for transmission on the innerloop. The output of a code may involve sequentially outputting a code that is expanded at each level of processing.

Sequential hierarchical output involves one module activating a set of modules at the next stage of processing. A module at level $N-1$ transmits a vector, loading three to five modules at level N; the modules in level N transmit sequentially, loading multiple modules at level $N+1$. The architecture for sequential output is the same as that for input (see Fig. 2). However, to accomplish sequential output, the sequences of control signals between the level controller and the modules must be altered. For output, the system must load the buffers in parallel and output sequentially.

Figure 9, showing a simulation of sequential output, illustrates the output of a sequence of motor movements to write the word *cat*. Assume that a

"An important feature of multilayered connection networks is that they build internal representations, such that codes similar in one representation can be very dissimilar at higher levels of representation (see Hinton, 1986; Ackley, Hinton & Sejnowski, 1985).

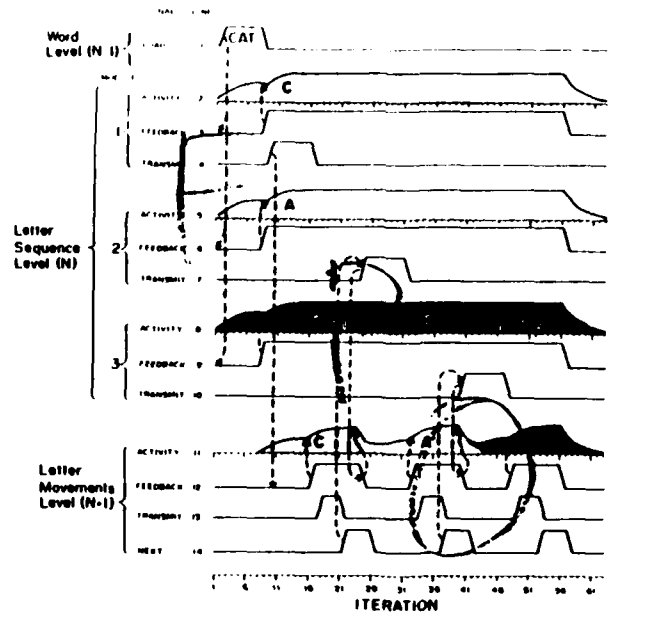

Fig. 9. Sequential output in CAP1 simulation. This diagram represents converting a code for the word *cat* to the individual motor sequences for Levels 4, 5, and 6 in Fig. 4. See Fig. 5 for detailed caption description. The "CAT" LOAD signal (line 1) causes parallel loading for the modules for each letter (lines 2, 3, 5, 6, 8, 9). These modules are then sequentially output by serially activating the TRANSMIT signals (lines 4, 7, 10) of the modules containing each letter. The sequential outputs load the next-level buffers, sequentially activating the letters *c*, *a*, and *t* (line 11). These messages are sequentially transmitted to the next level of processing (line 13). When the letter output module returns the third NEXT signal (line 14, iteration 52), the letter-sequence level clears its buffer and issues a NEXT signal to the previous level.

module in the lexical region transmits a code for the letter pattern of *cat* in the innerloop (see Fig. 3). In the motor system, the central controller first sets the feedback parameter to zero, thus clearing the contents of the buffer. Then the feedback is increased to latch the input for the pattern "CAT" in the module. Note that, since the module buffers the output code, other messages can be sent on the innerloop while the motor region is outputting

the "CAT" stimulus." After the multiple buffers "C," "A," "T" are loaded in parallel, feedback is maintained at a high level to maintain the traces. Level *N* now begins to sequentially output the active modules to level *N* + 1. Since the modules at a given level of processing do not interconnect, the modules within a level of processing can transmit their messages without distorting the information of neighboring modules. Sequentially activating the TRANSMIT control signals will sequentially output the contents of buffers.

The order of output can be determined in the same ways that sequential input can be maintained (see above). Potential methods for doing this include location-specific coding, e.g., Module 1 of the stage would always be the first out; context-sensitive coding, e.g., the module with a code indicator at the front of the list, code "-Ca", would be the first item out, and context would determine the next item, "cAt", then "aT-"; or strength-coding, e.g., the first module would have the highest strength and inhibit the gain control of all the other modules until it is output (see Rumelhart & Norman, 1982). Order could be determined by any of these methods within a level of processing. The module with the highest priority would inhibit the output of the other modules at its level and output its message, e.g., set the TRANSMIT signal to transmit the "C" code (Fig. 9, line 4), and the LOAD control signal to the *N* + 1 level of processing. Level *N* + 1 begins the same sequence of events as in level *N*. At level *N* + 1 the code of the "C" would be converted into the sequence of motor movements to produce the line strokes for the "C." When level *N* + 1 finished outputting all its active modules, it would send a NEXT signal (Fig. 9, line 14) to the level *N* controller requesting the next input. At level *N*, the next-highest priority module, e.g., "A," would be transmitted (Fig. 9, line 7). This process would continue until all the active modules at *N* had been output. Then level *N* would send a NEXT signal to level *N* − 1. If the module sending the NEXT signal were on the innerloop, the NEXT signal would be routed through the central controller to the module originating the transmission to the innerloop.

This sequential output scheme provides a robust method of outputting information. Should an error occur at any level of processing, the previous stage would have sufficient information to reload the next stage. A module would not clear its contents until the next level had indicated, for all the codes from the previous level, that the information was received, decoded,

[15]As with sequential input, latching input to a module by using feedback will block other dissimilar messages from distorting the code within the buffer. This implies that nonrelated messages can be transmitted on the innerloop. However, if related codes are transmitted, interference will result, e.g., in the Stroop task (see Dyer, 1973) both the print and color codes are transmitted; since these are similar codes, the feedback latching mechanisms will be distorted by these multiple transmissions.

and successfully transmitted to the next level; e.g., in Fig. 9 the "C" is not cleared until after level *N* + 1 reports back that the "T" code was successfully transmitted. This system is asynchronous, meaning that each stage can operate at its own temporal scale, where information at a previous stage of processing is buffered until it is needed at the next stage. If a later stage were to alter its output, e.g., pressing the shift key to type certain characters, the later level could take more or less time for each of its sequential outputs. An additional level of robustness is provided by the context-storage process of the innerloop modules. For example, assume an interrupt occurred, halting all output and flushing all the modules. Once the network resumed outputting sequentially, the context vector could be transmitted; this would allow the innerloop modules to be reloaded. The network could then begin to output by resuming activity at the point of the innerloop transmissions which preceded the last context-storage event.

The process of sequencing information is very similar throughout the system (see Fig. 4). In the input region, modules send a LOAD signal to the next level when information is ready for the next level of processing. The LOAD signal indicates to the next higher level that it should try to recognize (via the process of increasing feedback) a code incorporating the active input at the previous level. The higher input level sends back a NEXT control signal when it recognizes the total pattern of the previous level. The NEXT signal results in the next input to that level, flushing the information at the previous input so that new information can be loaded at that level of processing. In the output regions, each level sends a LOAD control signal to load a series of modules at the next level of processing. The next level returns a NEXT control signal when it has completed all the processing for the previous LOAD signal. The processing in the innerloop is similar, except that the source and destination of the control signals are not limited to a single set of modules. Within the input and output regions, the control LOAD and NEXT signals come from the next level in the same region. In the innerloop, the motor region may get input from any of the regions on the innerloop. The control signals must be routed through the central-control structure. The working memory within the central-control structure must maintain information indicating where to route the NEXT signal when it is issued by a module in the innerloop.[16]

[16]A simple implementation of the central-control routing might involve having the central controller passively monitor the traffic in the innerloop by using changes in activation to specify the intended routing path. For example, if the visual system were to transmit the code on the innerloop, the central-control monitoring would be able to detect the sequential change in activity in the visual region and the region that responded to the visual transmission. Assuming the motor system were activated by the "CAT" transmission, the central controller could infer the modules to which the visual system was outputting. Then, if the motor system were to send a NEXT signal, the central controller would route the NEXT signal to the visual region.

B. Chunking

The proposed architecture produces many of the chunking effects that Johnson (1966a, b, 1970, 1972) has described. Four phenomena are of special interest. First, subjects will naturally group input and output sequences in groups of three to four elements with longer pauses between groups. In the present model, codes for a given level of processing should not contain more information than the control level can handle, suggesting a need for grouping and increased delays when levels are reloaded. Second, the probability of outputting the first element of a sequence is dependent on the number of chunks at each level of processing, but not on the size of chunks other than the first one at each level of processing. To output a 3, 3, 3 sequence requires decoding three elements at the top level and three at the next level, or six altogether. To output a 3, 2, 2, 2 sequence requires outputting four elements at the top level and two or three at the next level, for a total of seven. Human recall of the first items of a nine-element list is better for a 2, 4, 3 than it is for a 3, 3, 3 or 3, 2, 2, 2 code.

In the present architecture, the first elements of every chunk must be output before the first bottom-level code produces output. A failure at any level will terminate the output process. However, the number of elements in unexpanded chunks should not influence the probability of output of the elements of a chunk, i.e., whether the next chunk at a level to be output codes two or five chunks should not influence the probability of output of the elements of the present chunk.

The third chunking phenomenon centers of the fact that subjects tend to pause longer between chunks than within chunks (see Broadbent, 1975, McLean & Gregg, 1967; Reitman & Rueter, 1980). This is illustrated in skilled memory studies in which SF outputs digits while performing the digit-span task. By analyzing SF's verbal protocols, Chase and Ericsson (1981) determined that his speech patterns nearly always followed the same pattern. Digit groups were recalled at a rate of about three digits/sec, with pauses of about 2 sec between groups. The processes of LOAD and NEXT that occur when one level transmits to the next level will produce longer pauses in the outputs. This would be the case particularly when innerloop transmissions are involved, due to time added waiting for other innerloop traffic to be stopped and the NEXT signal to be routed.

The fourth chunking phenomenon involves Johnson's (1970, 1972) characterization of a chunk as an "opaque container" that must be treated as a complete pattern at a given level of processing and not just as the concatenation of the codes of the previous level. According to Johnson (1970, p. 213), a container "is opaque in the sense that recovery from memory of the code container does not allow the S to evaluate the information he has recovered." Johnson found that if a subject learns multiple strings and

repeats elements of a chunk, but not the full chunk, accuracy does not improve. If, however, the full string or the first chunk is repeated, performance does improve; e.g., if one repeats the string 94 487 3587 then 39 687 3932, repeating the 87 3 sequence on every other list produces no greater recall than random digits. In the present architecture, the higher-level codes are encapsulated codes containing a distributed representation of the total information that is not divided into individual elements until the next level decodes it. If most of the learning occurs in the innerloop, there is little benefit for repeating portions of the lower-level codes.

In summary, the connectionist/control architecture can perform robust sequential output which exhibits many of the phenomena associated with serial output and chunking. Each level of processing buffers and encodes information. Control signals between levels, e.g., the NEXT and LOAD signals, provide a single mechanism for accounting for chunking effects in input, innerloop, and output processing.

VIII. Workload and Working Memory

The current architecture can perform multiple tasks concurrently. The system has a variety of resources that can be allocated in different ways to meet the demands of different task combinations. When multiple tasks compete for limited resources, processing will either be delayed or errors will result. This architecture includes many types of resources, e.g., buffers, regions, control structure, and connection weights, and contrasts sharply with Kahneman's (1973) proposal that attention is a single undifferentiated resource. The present architecture is consistent with Wicken's (1980) view that resources are differentiated by modalities. However, in addition to competition for specific regions as in Wicken's model, the present architecture emphasizes the importance of competition for the control structure. This architecture and simulation model are also used to account for human attention phenomena and the acquisition of component skills (see Schneider & Mumme, 1987). In the present section we limit our discussion to how the connectionist/control architecture can account for workload effects in memory tasks.

The connectionist/control architecture can employ five strategies to perform concurrent tasks. The first strategy is to buffer and delay messages for one of the tasks until the other task is completed. Recall that the system is asynchronous, with buffers at every level of processing. If two tasks require the same set of modules on the innerloop, the central controller can sequence the transmission on the innerloop to time share the use of critical modules. Since both the inputs and outputs are buffered, the time sharing generally results in longer reaction times, but not greater errors. Research

on the psychological refractory period (see Smith, 1967) illustrates such slowing. If the subject must respond to two stimuli presented successively, the response to the second stimulus is delayed by about the time required to complete the response to the first signal.

The second strategy is to move a task into low-use buffers. For example, if a subject were to maintain three digits in auditory buffers while performing a visual task that utilizes the visual system, both the innerloop and motor system would show little speed or accuracy deficits. Baddeley and Hitch (1974) have found that increasing the short-term digit load from one to three digits results in no change in accuracy and little change in speed of processing, e.g., a 5% (0.07 sec) slowing in judging sentences such as *Canaries have wings* to be true. However, loads that exceed the capacity of buffers result in substantial errors and increases in reaction time; e.g., with an eight-digit loading the Baddeley and Hitch (1974) sentence-judging task resulted in a substantial increase in errors (from 3% to 14%) and a slowing of the response (44%, 0.67 sec).

The third strategy to deal with high workload is to use context storage to temporarily associate information to the current context and utilize the context to load modules. The ability of subjects to perform embedded tasks after a brief rehearsal period suggest this type of strategy. For example, Klapp *et al.* (1983) allowed subjects 5 sec to rehearse letter strings 0, 6, and 9 items long before they performed an embedded task such as visual scanning. In the connectionist/control architecture, the short rehearsal would associate the letters to the context, then the search task could be performed without rehearsing the letter task, and finally the context could be used to retrieve the letters. This context-storage strategy can explain the use of brief review periods before performing critical events. For instance, in both athletic competition and military combat situations, individuals often review their intended actions just prior to entering the critical situation. This process of review could be used to associate the impending actions to the context. Attending to the context, i.e., transmitting the context vector, could then simultaneously load modules in many regions and initiate many concurrent processes.

The fourth strategy is to develop automatic processes to reduce the load on the central and regional controllers. We assume that the control-processing system can control only a very small proportion of the modules in the network. The regional controllers generally only buffer three to four elements at a level of processing. To reduce the load on the control architecture, each module can gate information locally. A model for the development of local automatic gating is detailed in Schneider and Mumme (1987). Briefly, they assume that the autoassociation matrix within each module associates *the message within the module with a priority tag.* Transmissions from the module that result in a positive event (determined at the system

level) increase the priority tag, negative transmissions decrease it. If a module receives a high-priority message, the module transmits the message in the absence of control input. If the system consistently responds to particular messages, those messages will be automatically transmitted, i.e., as a result of the local priority tag. The benefit of priority-based transmission is is that is allows the limited control-processing resources to be used elsewhere in the system. The model of priority-tag learning (Schneider & Mumme, 1987) illustrates how consistent practice develops fast, parallel, and difficult-to-alter automatic processing.

The fifth strategy for dealing with high workload is to reduce the message interference for concurrently transmitted messages. Message interference is a limiting factor for communications on the innerloop. Each module on the innerloop has its own fibers, allowing multiple messages to be transmitted concurrently. However, if two incoming messages activate competing vectors in a receiving module, interference results. Assume a typist seeks to perform copy typing while concurrently comprehending conversations. Normally the visual transmission of text codes activates semantic processing (for comprehension as in reading) and motor processing (for typing). The auditory transmission of speech codes normally activates semantic processing and articulatory codes. Initially concurrent auditory and speech input cause interference, and the central control system only allows the transmission of visual codes during typing. As the subject practices typing, i.e., transmitting messages from the visual system and releasing messages in the motor system, the visual-to-motor connections are strengthened. The lack of releasing responses in the comprehension system weakens these connections. *With time, the visual-to-semantic connections weaken such that* visual input no longer interferes (at least in a typing context) with the auditory input to semantic processing. If the visual transmissions become automatic, the central controller need not be involved in copy typing. At this stage, the typist could attend to the auditory input and comprehend speech while typing. Copy typists' lack of memory for the material typed is suggestive of this kind of change of connection weights.

IX. Working Memory in Learning and Skill Acquisition

Working memory plays a critical role in learning and acquiring knowlege. All LTM is stored in the connection weights in the network. The change in connection weights is determined by what is active in working memory. In the process of learning a task, controlled processing is generally used to compare the input pattern to a rule and to perform the appropriate response based on the match. One could view this as a process of acquiring productions (J. R. Anderson, 1983). However, since many patterns are stored in

any single connection matrix, there will be interactions among patterns, depending on the total set of productions to be acquired (see Rumelhart & McClelland, 1986a).

Acquiring a skill necessitates keeping instructions and task-relevant information in working memory while performing at least some components of the task. For example, to learn to specify the output of an AND electronic gate, the system must store the verbal rule "*if all the inputs are high, then* the output is high," activate the input patterns, compare the input patterns to the pattern "all high," and respond "high output" if true and "low output" if false. The first step of skill acquisition is to rehearse the verbal rule to enable the context to load the buffers. The context would preload modules for the target state (e.g., a high on all inputs), the response on a match (e.g., a prediction of a high on its output), and the response on a nonmatch (e.g., prediction of a low on the output). By associating these patterns to the context vector with fast weights, the context could be revoked to reload the buffers. If the subject were distracted, the instructions could be reloaded by activating the context vector. When a problem, (e.g., What is the output *if the input is 1111?) is presented, a controlled* comparison would occur between the input and the target output. On a match the "yes" response would be released. As a result of controlled processing operations, the input pattern, e.g., 1111, would be transmitted, followed by the output pattern, i.e., a high response, being transmitted. This would associate the input to the output. With sufficient training trials, the long-term connections between the input and output would be modified such that the input could directly activate the output (see Schneider & Mumme, 1987, for a simulation for such learning). When this occurs, context preloading and the controlled comparison process are not needed.

A. Distributing Practice

The importance of context storage for learning to perform a task raises serious issues concerning how problems should be sequenced and spaced. Initially it is beneficial to mass practice on a task. For example, in learning electronic troubleshooting, it is better to start with a block of trials for a single gate type before moving on to the next gate type. This is preferred because the context storage maintains the working memory. In procedural tasks, subjects learn to perform individual procedures quickly during massed practice of single tasks, but then show poor performance when the trial types are intermixed. Due to PI between codes, context cannot be used to maintain or retrieve codes when training is distributed. Hence, more errors are expected during distributed training than massed training.

To be able to perform a variety of procedures in random order, training must progress to distributing practice. The marginal utility of massed practice decreases with time. If the context vectors eliminate most of the error between the activated output and desired output, there is less learning (with a delta-type learning rule). Also, if the subject must randomly execute the procedures at different times, context-based learning may show poor transfer. In sum, the advantages of massing practice early to maintain information in working memory trades off against the disadvantages of the context learning, showing poor transfer and reducing long-term learning. Procedures which expand the distribution of practice with training are likely to be optimal (Landauer & Bjork, 1978).

B. Phases of Skill Acquisition

Within the present architecture, there are five identifiable phases of skill acquisition. The movement between these phases is a gradual, continuous transition. The use of working memory and controlled processing varies at each phase of processing. The rate of movement between stages depends on the nature of the task to be learned. We illustrate the transitions using numbers based on subjective impressions of learning logic gates for electronic troubleshooting (Carlson & Schneider, 1987). These numbers are included only to give the reader an impression of the expected time course of these changes.

Phase One of skill acquisition, e.g., Trials 1–4, involves loading all the information for performing the task into buffers. The task is performed by comparing information in the buffers to the incoming stimuli and releasing a response if a match occurs (see Schneider, 1985; Schneider & Mumme, 1987, for details). If the subject is interrupted, the buffer information may be lost, resulting in errors. We train our subjects with a mini lecture on six gate types. In learning logic gates, our subjects' response times are between 2–3 sec on the first trial, with subjects requesting help about 40% of the time.

Phase Two of skill acquisition, e.g., Trials 5–20, involves performing the same task as Phase One, but by Trial 5 the context-storage mechanism can maintain and reload working memory. Performance for blocked trials of the task is accurate and reasonably fast. By Trial 5, subjects' accuracy is near perfect and response times are down to 0.7 seconds for massed trials. During massed practice in Phase 2, controlled processing resources are required to compare the input to the rules and to release output vectors, but they are not necessary to maintain the traces in the buffer. However, if alternative procedures are intermixed, accuracy decreases and responding slows considerably. Whenever the task switches, subjects reevoke the verbal rule and context to reload the buffers in order to perform the task.[1]

[1] This is similar to J. R. Anderson's (1983) use of interpretive execution of productions

On early intermixed trials, subjects' response times increase to 2-3 sec and they request help about 40% of the time.

Phase Three of skill acquisition, e.g., Trials 21-100, occurs when the associations to the goal state are strong enough to load working memory without the use of context storage, such that attending to an AND gate loads the input pattern to be checked and the possible output responses. In this phase performance is accurate and rapid even if tasks are intermixed. However, the subject must still attend to the task and perform control-process comparisons.

In Phases Four and Five of skill acquisition, a substantial reduction occurs in the use of controlled processing resources in performing the task. Phase Four, e.g., 101-200 trials, is identified when the associations between input, the goal state, and the output become strong enough for the input to evoke the output directly; e.g., with an input of 111, and a goal of AND, the output would evoke a 1 output via associative retrieval. In this phase, controlled processing comparison drops out, thus reducing workload (see Schneider & Mumme, 1987, for simulation). Note that controlled processing is still required to transmit messages on the innerloop and to route the NEXT and LOAD signals. In learning electronic troubleshooting, subjects show small speedups (e.g., 100 msec in predicting the output of single gates) between 100-200 trials of practice, but dramatic speedups (from 8 to 4 sec) in problem solving in circuit troubleshooting. This improved ability to use the rule in the problem-solving context suggests that the learning during Phase Four eliminates the control-processing comparisons as in Phases One to Three. Phase Five, e.g., after about 200 trials/rule, occurs when the modules develop local automatic processing so that the message is transmitted even in the absence of controlled processing input. At this phase of processing, controlled processing resources need not be allocated in the gate identification task. The task can be performed reliably even if the subject uses controlled processing resources to perform other tasks. Some alternative tasks do interfere due to message interference.

In the connectionist/control architecture, the extent to which working memory is used varies, depending on the task and the phase of skill acquisition. The combination of context storage and controlled process comparison enables the network to accurately perform novel tasks after only a few trials. This contrasts with traditional connectionist learning systems that typically require thousands of trials to acquire a novel set of associations (see Schneider, 1987). The first few trials of performing a task are very attention demanding, difficult to perform in mixed trials, and error prone under high workload. With practice, the system modifies the LTM associations such that automatic processing develops which enables fast, accurate, and low-resource load processing.

X. Final Comments

The connectionist/control architecture details a computational system that exhibits many of the phenomena of human working memory. The system level of the architecture (see Fig. 3) includes regions that specialize in different classes of processing. The activity of the regions is coordinated by a central control structure that routes control signals and sequences transmissions among regions to limit problems of message interference. One of the regions serves as a context storage mechanism that can [illegible] (via fast-weight connections) messages [illegible] the innerloop of processing. Each region is divided into a number of levels that sequentially or spatially input or output patterns to other levels (see Fig. 2). Each level has a control structure that monitors the activity of all the modules in its level and controls the feedback and transmission of that level. The level control structure sends and receives control signals to coordinate the sequential storage and processing of information. Each level includes multiple modules (see Fig. 1 and 2). Each of these modules involves a connectionist network that processes vectors of information. A module can store, categorize, maintain, and prioritize a received vector. This architecture is sufficiently detailed that it can simulate a wide variety of human learning and attentional phenomena. The architecture is physiologically plausible and shows some intriguing parallels to modular systems in the cortex (see Schneider & Mumme, 1987).

Any model of human working memory must first be evaluated as to whether it provides a robust processing architecture that could survive in the complex and dynamic world in which humans have evolved. Buffers are needed because much of the processing must be sequential and asynchronous. Attention is needed to deal with resource competition and message interference. A context-storage mechanism is needed to allow recovery from interruptions, to increase the effective size of working memory, and to permit acquisition of rudimentary skills after only a single stimulus presentation.

We think that the traditional models of working memory, e.g., Atkinson and Shiffrin (1968) and Baddeley (1986), do not provide a robust processing architecture. These buffer-oriented systems to not provide mechanisms to allow information to be recovered after an interruption that flushes the buffers. They provide a limited model for a subset of working-memory phenomena. A system limited to only such buffer memories and a slowly changing LTM is likely to exhibit severely unstable processing, [illegible] perhaps similar to the amnesiac patient HM. The buffer models do account for classic STM phenomena, e.g., interference effects. However, they do not account for many other important phenomena, e.g., lack of STM decay on the first trial, PI effects, reliable processing despite severe loading, and the critical dependence on LTM for what can be stored in STM.

We have described an architectural class of models for working memory. There are many possible configurations of modules, levels, regions, and control structures. For example, the innerloop of processing might be a ring as depicted in Fig. 3, or it could be some complex lattice of processing regions. A great deal of theoretical and simulation work needs to be performed to determine the computational capacities of this architecture. Human empirical research is required to (1) evaluate how well models within this architecture predict human data, and (2) identify specific details of the architecture.

The present architecture can account for a wide range of human working-memory phenomena as emergent properties of the system. Most of the predictions follow from the process of developing a robust processing system, rather than from trying to model specific phenomena. The proposed multileveled buffer scheme provides an interpretation of the magic number three or four, acoustic confusions, sequential processing, problems with digit canceling and reverse digit span, the difficulty of maintaining order information, and the nature of rehearsal. Context storage is included to enable the system to cope with interruptions and to expand working memory. This storage mechanism provides a way of interpreting the distinction between episodic and semantic memory, retroactive and proactive interference effects and trade offs, the buildup of PI, the benefit of elaborative rehearsal over maintenance rehearsal, the release of proactive interference either by time or switching content, LTM recency effects, and the ability to continue processing information after traditional STM capacity is exceeded.

The present processing architecture can be operated with different levels of effectiveness depending on how the resources in the system are utilized. The skilled uses of working memory provide interpretations of the unattended speech effect, levels of processing, mnemonics, category clustering, distribution of practice, generation effects, and skilled memory. The control processing for sequential output of information makes predictions regarding chunking, chunk-based retrieval, and pause boundaries. The control processing management of information enables the system to deal with conditions of high workload and produces psychological refractory-period phenomena, sequential attending, and the use of context to facilitate priming. Context storage enables information to be acquired rapidly during massed practice of procedures and illustrates that using an expanding practice schedule results in better retention for later distributed testing. To reduce workload on the limited control processing system, the control of information is localized within modules. This localization takes place gradually and illustrates different phases of skill acquisition.

This architecture represents a hybrid of many previous models and frameworks for memory. It includes buffers (Waugh & Norman, 1965; Atkinson & Shiffrin, 1968), a system to perform automatic and controlled processing (Shiffrin & Schneider, 1977; Schneider, 1985; Schneider & Mumme, 1987), multiple processing regions (Baddeley, 1976; Wickens, 1970, 1972), a distributed connectionist approach to associative memory (McClelland & Rumelhart, 1986; Rumelhart & McClelland, 1986a), auto-associative categorization (J. A. Anderson, 1983), automatic context storage (Tulving, 1972, 1983, 1984; Raaijmakers & Shiffrin, 1980, 1981; Hasher & Zacks, 1979), and the use of fast connection weights (Hinton & Plaut, 1987).

The understanding of working memory is critical to the understanding of human cognition. We must know its capacity, structure, strategies of use, and limitations. It is important to examine a variety of architectures that incorporate the complex diversity of working-memory phenomena seen in humans. The present connectionist/control architecture provides a potential architecture that could be implemented in a physiologically feasible manner and predicts a variety of the phenomena and potential structure of human memory.

Acknowledgments

This research was sponsored by the Army Research Institute, under Contract No. MDA903-86-C-0149 and Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research under Contract Nos. N-0014-86-K-0107 and N-0014-86-K-0678.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9, 147–169.

Anders, T. R., & Lillyquist, T. D. (1971). Retrieval time in forward and backward recall. *Psychonomic Science*, 22, 205–206.

Anderson, J. A. (1983). Cognitive and psychological computation with neural models. *IEEE Transactions on System[illegible] Man, and Cybernetics*, SMC-13, 799–815.

Anderson, J. A., & Mozer, M. C. (1981). Categorization and selective neurons. In G. E. Hinton & J. A. Anderson (Eds.), *Parallel models of associative memory*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Atkinson, R. C., & Shiffrin, R. M. (1968). Human memory: A proposed system and its control process. In K. W. Spence & J. T. Spence (Eds.), *The psychology of learning and motivation. Vol. 2*. New York: Academic Press.

Atkinson, R. C., & Shiffrin, R. M. (1971). The control of short-term memory. *Scientific American*, 224, 82–90.

Ayres, T. J., Jonides, J., Reitman, J. S., Egan, J. C., & Howard, D. A. (1979). Differing suffix effects for the same physical stimulus. *Journal of Experimental Psychology: Human Learning and Memory*, 5, 315-321.

Baddeley, A. D. (1966). Short-term memory for word sequences as a function of acoustic, semantic, and formal similarity. *The Quarterly Journal of Experimental Psychology*, 18, 362-365.

Baddeley, A. D. (1976). *The psychology of memory*. New York: Basic Books.

Baddeley, A. D. (1983). Working memory. *Philosophical Transactions of the Royal Society of London, Series B*, 302, 311-324.

Baddeley, A. D. (1986). *Working memory*. Oxford: Clarendon.

Baddeley, A. D., Grant, S., Wight, E., & Thomson, N. (1974). Imagery and visual working memory. In P. M. A. Rabbitt & S. Dornic (Eds.), *Attention and performance VIII*. Hillsdale, NJ: Erlbaum.

Baddeley, A. D., & Hitch, G. (1974). Working memory. In G. H. Bower (Ed.), *The psychology of learning and motivation. Vol. 8*. New York: Academic Press.

Baddeley, A. D., & Hitch, G. J. (1977). Recency reexamined. In S. Dornic (Ed.), *Attention and performance VI*. Hillsdale, NJ: Erlbaum.

Baddeley, A. D., & Scott, D. (1971). Short-term forgetting in the absence of proactive inhibition. *The Quarterly Journal of Experimental Psychology*, 23, 275-283.

Barnard, P. (1985). Interacting cognitive subsystems: A psycholinguistic approach to short-term memory. In A. W. Ellis (Ed.), *Progress in the psychology of language. Vol. 2*. Hillsdale, NJ: Erlbaum.

Barrett, E. F., & Mangleby, K. L. (1976). Physiology of cholinergic transmission. In A. M. Goldberg & I. Hanin (Eds.), *Biology of cholinergic function* (pp. 29-100). New York: Raven.

Bellezza, F. S. (1981). Mnemonic devices: Classification, characteristics, and criteria. *Review of Educational Research*, 51, 247-275.

Bousfield, W. A. (1953). The occurrence of clustering in the recall of randomly arranged associates. *Journal of General Psychology*, 49, 229-240.

Bousfield, W. A., & Cohen, B. H. (1955). The occurrence of clustering in the recall of randomly arranged words of different frequencies of use. *Journal of General Psychology*, 52, 83-95.

Bousfield, W. A., & Cohen, B. H. (1956). Clustering in recall as a function of the number of word categories in stimulus word lists. *Journal of General Psychology*, 54, 95-106.

Bower, G. H. (1970). Analysis of a mnemonic device. *American Scientist*, 58, 496-510.

Bower, G. H., Clark, M. C., Lesgold, A., & Winzenz, D. (1969). Hierarchical retrieval schemes in recall of categorized word lists. *Journal of Verbal Learning and Verbal Behavior*, 8, 323-343.

Broadbent, D. E. (1958). *Perception and communication*. Oxford: Pergamon.

Broadbent, D. E. (1975). The magic number seven after fifteen years. In A. Kennedy & A. Wilkes (Eds.), *Studies in long-term memory*. New York: Wiley.

Broadbent, D. E. (1984). The Maltese cross: A new simplistic model for memory. *The Behavioral and Brain Sciences*, 7, 55-94.

Brown, J. (1958). Some tests of the decay theory of immediate memory. *The Quarterly Journal of Experimental Psychology*, 10, 12-21.

Carlson, R. A., & Schneider, W. (1987). *Learning and using causal rules*. Unpublished manuscript.

Chase, W. G., & Ericsson, K. A. (1981). Skilled memory. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, NJ: Erlbaum.

Chase, W. G., & Ericsson, K. A. (1982). Skill and working memory. In G. H. Bower (Ed.), *The psychology of learning and motivation. Vol. 16*. New York: Academic Press.

Bellezza, F. S. (1982). Improve your memory skills. Englewood Cliffs, NJ: Prentice-Hall.

Cohen, N. J., & Squire, L. R. (1980). Preserved learning and retention of pattern analyzing skill in amnesia: Dissociation of knowing how and knowing that. *Science*, 210, 207-210.

Cohen, N. J., Eichenbaum, H., Deacedo, B. S., & Corkin, S. (1985). Different memory systems underlying acquisition of procedural and declarative knowledge. *Annals of the New York Academy of Sciences*, 444, 54-71.

Conrad, R. (1959). Errors of immediate memory. *British Journal of Psychology*, 50, 349-359.

Conrad, R. (1960). Serial order intrusions in immediate memory. *British Journal of Psychology*, 51, 45-48.

Conrad, R. (1964). Acoustic confusions in immediate memory. *British Journal of Psychology*, 55, 75-84.

Crannell, C. W., & Parrish, J. M. (1957). A comparison of immediate memory span for digits, letters, and words. *Journal of Psychology*, 44, 319-327.

Crowder, R. G. (1982). The demise of short-term memory. *Acta Psychologica*, 50, 291-323.

Crowder, R. G., & Morton, J. (1969). Precategorical acoustic storage (PAS). *Perception & Psychophysics*, 5, 365-373.

Cuddy, L. J., & Jacoby, L. L. (1982). When forgetting helps memory: An analysis of repetition effects. *Journal of Verbal Learning and Verbal Behavior*, 21, 451-467.

Desimone, R., Schein, S. J., Moran, J., & Ungerleider, L. G. (1985). Contour, color and shape analysis beyond the striate cortex. *Vision Research*, 25, 441-452.

Dillani, D. K. (1972). Most efficient chunk sizes. *Cognitive Psychology*, 3, 355-359.

Dyer, F. H. (1973). The Stroop phenomenon and its use in the study of perceptual, cognitive and response processes. *Memory & Cognition*, 1, 106-120.

Ericsson, K. A., & Chase, W. G. (1981). Exceptional memory. *American Scientist*, 70, 607-615.

Ericsson, K. A., & Polson, P. G. (1987). A cognitive analysis of exceptional memory for restaurant orders. In M. T. H. Chi, R. Glaser, & M. J. Farr (Eds.), *The nature of expertise*. Hillsdale, NJ: Erlbaum, in press.

Estes, W. K. (1972). An associative basis for coding and organization in memory. In A. W. Melton & E. Martin (Eds.), *Coding processes in human memory*. Washington, DC: Winston.

Fisk, A. D., & Schneider, W. (1984). Memory as a function of attention, level of processing, and automatization. *Journal of Experimental Psychology. Learning, Memory & Cognition*, 10, 181-197.

Frick, R. W. (1984). Using both an auditory and a visual short-term store to increase digit span. *Memory & Cognition*, 12, 507-514.

Goggin, J., & Wickens, D. O. (1971). Proactive interference and language change in short term memory. *Journal of Verbal Learning and Verbal Behavior*, 10, 453-458.

Hasher, L., & Zacks, R. T. (1979). Automatic and effortful processes in memory. *Journal of Experimental Psychology: General*, 108, 356-388.

Healy, A. F. (1974). Separating item from order information in short-term memory. *Journal Verbal Learning and Verbal Behavior*, 13, 644-655.

Healy, A. F. (1982). Short-term memory for order information. In G. H. Bower (Ed.), *The psychology of learning and motivation. Vol. 16*. New York: Academic Press.

Hinton, G. E. (1986). Learning distributed representation of concepts. *[8th Annual Conference of the Cognitive Science Society* pp. 1-12. Amherst, MA, August 15-17.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing. Vol. 1*. Cambridge, MA: MIT Press.

Hinton, G. E., & Plaut, D. C. (1987). *Using fast weights to deblur old memories*. Ninth Annual Conference of the Cognitive Science Society, Seattle, Washington, July, 1987.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann Machines. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing. Vol. 1*. Cambridge, MA: MIT Press.

James, W. (1890/1983). *The principles of psychology*. Cambridge, MA: Harvard University Press.

Jarvella, R. J. (1971). Syntactic processing of connected speech. *Journal of Verbal Learning and Verbal Behavior*, 10, 409-416.

Johnson, N. F. (1966a). The influence of associations between elements of structured verbal responses. *Journal of Verbal Learning and Verbal Behavior*, 5, 369-374.

Johnson, N. F. (1966b). On the relationship between sentence structure and the latency in generating the sentence. *Journal of Verbal Learning and Verbal Behavior*, 5, 375-380.

Johnson, N. F. (1970). The role of chunking and organization in the process of recall. In G. H. Bower (Ed.), *The psychology of learning and motivation. Vol. 4*, New York: Academic Press.

Johnson, N. F. (1972). Organization and the concept of a memory code. In A. W. Melton & E. Martin (Eds.), *Coding processes in human memory*. Washington, DC: Winston.

Just, M. A., & Carpenter, P. A. (1987). *The psychology of reading and language comprehension*. Boston: Allyn and Bacon, Inc.

Kahneman, D. (1973). *Attention and effort*. Englewood Cliffs, NJ: Prentice-Hall.

Keppel, G., & Underwood, B. J. (1962). Proactive inhibition in short-term retention of single items. *Journal of Verbal Learning and Verbal Behavior*, 1, 153-161.

Kincaid, J. P., & Wickens, D. D. (1970). Temporal gradient of release from proactive inhibition. *Journal of Experimental Psychology*, 86, 313-316.

Kirchner, W. K. (1958). Age differences in short-term retention of rapidly changing information. *Journal of Experimental Psychology*, 55, 352-358.

Klapp, S. T. (1987). Short-term memory limits in human performance. In P. Hancock (Ed.), *Human factors psychology*. Amsterdam: North-Holland.

Klapp, S. T., Marshburn, E. A., & P. T. Lester (1983). Short-term memory does not involve the "working memory" of information processing: The demise of a common assumption. *Journal of Experimental Psychology: General*, 112, 240-264.

Klapp, S. T., & Philliposff, A. (1983). Short-term memory limits in performance. In A. T. Pope & L. D. Haugh (Eds.), *Proceedings of the human factor society 27th annual meeting*. Santa Monica, CA: Human Factors Society.

Kohonen, T. (1984). *Self-organization and associative memory*. New York: Springer-Verlag.

Laird, J., Rosenbloom, P., & Newell, A. (1986). *Universal subgoaling and chunking: The automatic generation and learning of goal hierarchies*. Boston, MA: Kluwer.

Landauer, T. K., & Bjork, R. A. (1978). Optimum rehearsal patterns and name learning. In M. M. Gruneberg, P. E. Morris, & R. N. Sykes (Eds.), *Practical aspects of memory*. London: Academic Press.

Lashley, K. S. (1951). The problem of serial order in behavior. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior*. New York: Wiley.

Loess, H. (1968). Short-term memory and item similarity. *Journal of Verbal Learning and Verbal Behavior*, 7, 87-92.

Loess, H., & Waugh, N. C. (1967). Short-term memory and inter-trial interval. *Journal of Verbal Learning and Verbal Behavior*, 6, 455-460.

Logan, G. D. (1979). On the use of concurrent memory load to measure attention and automaticity. *Journal of Experimental Psychology: Human Perception and Performance*, 5, 189-207.

Lyon, D. (1977). Individual differences in immediate serial recall: A matter of mnemonics? *Cognitive Psychology*, 9, 403-411.

Mackworth, J. (1959). Paced memorizing in a continuous task. *Journal of Experimental Psychology*, 58, 206-211.

Mandler, G. (1967). Organization and memory. In K. W. Spence & J. T. Spence (Eds.), *The psychology of learning and motivation. Vol. 1*. New York: Academic Press.

McClelland, J. L., & Rumelhart, D. E. (1986). A distributed model of human learning and memory. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), *Parallel distributed processing. Vol. 2: Psychological and biological models*. Cambridge, MA: MIT Press.

McConkie, G. W., & Zola, D. (1979). Is visual information integrated across successive fixations in reading? *Perception & Psychophysics*, 25, 221-224.

McLean, R. S., & Gregg, L. W. (1967). Effects of induced chunking on temporal aspects of serial recitation. *Journal of Experimental Psychology*, 74, 455-459.

Melton, A. W. (1963). Implications of short-term memory for a general theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 2, 1-21.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63, 81-97.

Mountcastle, V. B. (1979). An organizing principle for cerebral function: The unit module and the distributed system. In F. O. Schmitt & F. G. Worden (Eds.), *The neurosciences*. Cambridge, MA: MIT Press.

Murdock, B. B., Jr. (1961). The retention of individual items. *Journal of Experimental Psychology*, 62, 618-625.

Peterson, L. R., & Gentile, A. (1965). Proactive interference as a function of time between tests. *Journal of Experimental Psychology*, 70, 473-478.

Peterson, L. R., & Peterson, J. J. (1959). Short-term retention of individual verbal items. *Journal of Experimental Psychology*, 58, 193-198.

Postman, L., & Phillips, L. W. (1965). Short-term temporal changes in free recall. *The Quarterly Journal of Experimental Psychology*, 17, 132-138.

Raaijmakers, J. G. W., & Shiffrin, R. M. (1980). SAM: A theory of probabilistic search of associative memory. In G. H. Bower, (Ed.), *The psychology of learning and motivation. Vol. 14*. New York: Academic Press.

Raaijmakers, J. G. W., & Shiffrin, R. M. (1981). Search of associative memory. *Psychological Review*, 88, 93-134.

Reisberg, D., Rappaport, I., & O'Shaughnessy, M. (1984). Limits of working memory: The digit digit-span. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 10, 203-221.

Reitman, J. S., & Rueter, H. H. (1980). Organization revealed by recall orders and confirmed by pauses. *Cognitive Psychology*, 12, 554-581.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical conditioning II: Current theory and research*. New York: Appleton.

Rosenberg, C. R., & Sejnowski, T. J. (1986). The spacing effect on NETtalk, a massively-parallel network. *The 8th Annual Conference of the Cognitive Science Society* pp. 72-89.

Ross, B. H. (1984). Remindings and their effects in learning a cognitive skill. *Cognitive Psychology*, 16, 371-416.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.) *Parallel distributed processing. Vol. 1*. Cambridge, MA: MIT Press.

Rumelhart, D. E., & McClelland, J. L. (1986a). On learning the past tense of English verbs. In J. L. McClelland & D. E. Rumelhart (Eds.), *Parallel distributed processing. Vol. 2: Psychological and biological models*. Cambridge, MA: MIT Press.

Rumelhart, D. E., & McClelland, J. L. (Eds.). (1986b). *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1: Foundations*. Cambridge, MA: MIT Press.

Rumelhart, D. E., & Norman, D. A. (1982). Simulating a skilled typist: A study of skilled cognitive-motor performance. *Cognitive Science* 6, 1-36.

Mishkin, M., & Appenzeller, T. (1987). The anatomy of memory. Scientific American, 256, 80-89.

Mishkin, M., Malmut, B., & Bachevalier, J. (1984). Memories and habits: Two neural systems. In G. Lynch, L. McGaugh, & N. M. Weinberger (Eds.), Neurobiology of learning and memory. New York: The Guilford Press.

Salame, P., & Baddeley, A. (1982). Disruption of short-term memory by unattended speech: Implications for the structure of working memory. *Journal of Verbal Learning and Verbal Behavior*, 21, 150-164.

Schank, R. C. (1982). *Dynamic memory*. New York: Cambridge University Press.

Schneider, W. (1985). Toward a model of attention and the development of automatic processing. In M. I. Posner & O. S. M. Marin (Eds.), *Attention and performance XI*. Hillsdale, NJ: Erlbaum.

Schneider, W. (1987). *Connectionism: Is it a paradigm shift for psychology? Behavior Research Methods, Instruments & Computers*, 19, [illegible]. 75-83

Schneider, W. S., & Desimone, R. (1986). *A combined physiological and theoretical approach to the mechanism of selective attention*. Unpublished paper.

Schneider, W., & Mumme, D. (1987). *Attention, automaticity and the capturing of knowledge: A two-level cognitive architecture*. Manuscript for submission.

Shiffrin, R. M., & Schneider, W. (1977). Controlled and automatic human information processing. II. Perceptual learning, automatic attending and a general theory. *Psychological Review*, 84, 127-190.

Siggins, G. R., & Gruol, D. L. (1986). Mechanisms of transmitter action in the vertebrate central nervous system. In V. B. Mountcastle, F. E. Bloom, & S. R. Geiger (Eds.), *Handbook of physiology: The nervous system IV* (pp. 1-114). Bethesda, MD: American Physiological Society.

Slak, S. (1970). Phonemic recoding of digital information. *Journal of Experimental Psychology*, 86, 398-406.

Smith, M. C. (1967). Theories of the psychological refractory period. *Psychological Bulletin*, 67, 202-213.

Starr, A. S. (1929). The significance of the ratio maintained between the forward, reverse and rhythmic memory span as obtained in three-thousand individual examinations. *Psychological Bulletin*, 26, 172-173.

Sternberg, S. (1966). High speed scanning in human memory. *Science*, 153, 652-654.

Szentagothai, J. (1979). Local neuron circuits of the neocortex. In F. O. Schmidt & F. G. Worden (Eds.), *The neurosciences fourth study program*. Cambridge, MA: MIT Press.

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving & W. Donaldson (Eds.), *Organization of memory*. New York: Academic Press.

Tulving, E. (1983). *Elements of episodic memory*. Oxford: Clarendon.

Tulving, E. (1984). Precis of Elements of episodic memory. *The Behavioral and Brain Sciences*, 7, 223-268.

Tzeng, O. J. L. (1973). Positive recency effect in delayed free recall. *Journal of Verbal Learning and Verbal Behavior*, 12, 436-439.

Van Essen, D. C. (1985). Functional organization of primate visual cortex. In A. Peters & E. G. Jones (Eds.), *The cerebral cortex, Vol. 3*, New York: Plenum.

Waugh, N. C., & Norman, D. A. (1965). Primary memory. *Psychological Review*, 72, 89-104.

Welford, A. T. (1968). *Fundamentals of skill*. London: Methuen.

Wickelgren, W. A. (1964). Size of rehearsal group and short-term memory. *Journal of Experimental Psychology*, 68, 413-419.

Wickelgren, W. A. (1967). Rehearsal grouping and hierarchical organization of serial position cues in short-term memory. *The Quarterly Journal of Experimental Psychology*, 19, 97-102.

Wickelgren, W. A. (1969). Context-sensitive coding, associative memory, and serial order in (speech) behavior. *Psychological Review*, 76, 1-15.

Wickens, C. D. (1980). The structure of attentional resources. In R. Nickerson (Ed.), *Attention and performance VIII*. Hillsdale, NJ: Erlbaum.

Wickens, D. D. (1970). Encoding categories of words: An empirical approach to meaning. *Psychological Review*, 77, 1-15.

Wickens, D. D. (1972). Characteristics of word encoding. In A. W. Melton & E. Martin (Eds.), *Coding processes in human memory*. Washington, DC: Winston.

Schneider, W., & Detweiler, M. (1987). The role of practice in dual-task performance: Workload modelling in a connectionist/control architecture. To appear in Human Factors, Spring, 1988.